APTEAN

ONYX

OEP Administration Guide

Version No: 7.8

February 2017

4325 Alexander Drive, Suite 100 • Alpharetta, GA 30022-3740 • www.aptean.com • info@aptean.com

Copyright © 2017 Aptean. All Rights Reserved. These materials are provided by Aptean for informational purposes only, without representation or warranty of any kind, and Aptean shall not be liable for errors or omissions with respect to the materials. The only warranties for Aptean products and services are those set forth in the express warranty statements accompanying such products and services, if any, and nothing herein shall be construed as constituting an additional warranty. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express written permission of Aptean. The information contained herein may be changed without prior notice. Some products marketed by Aptean contain proprietary software components of other software vendors. Aptean and other Aptean products and services referenced herein as well as their respective logos are registered trademarks or trademarks of Aptean or its affiliated companies.

Contents

1

Introduction

Overview

This guide is intended for administrators and developers who configure, customize, and maintain the Onyx system.

After you have followed the instructions in the Onyx 7.8 Upgrade Guide and installed Onyx to your development environment, you can begin configuring and customizing the system to suit your organization's requirements.

- Event Management Framework - Onyx Integration
- Appointments and Notifications
- Navigator

Getting Started

This book guides you through the main steps for modifying OEP to suit your business needs, deploying OEP to your users, and administering and maintaining OEP for your users.

1. **Install the Onyx Demo Database**. Installing the demonstration database gives you insight into how OEP works and, because it provides sample data for all OEP features, enables you to explore all feature areas of OEP.
2. **Explore OEP**. It's a good idea to familiarize yourself with OEP and its many features before you begin to modify it.
3. **Plan your OEP implementation**. A well planned implementation is a successful implementation.
4. **Add domain data**. Add domain data to your OEP system.
5. **Configure your development environment**. Create a separate OEP profile for each set of OEP user interfaces that you plan to configure, and create user accounts for your UCW designers.
6. **Modify OEP**. Once you understand how your various groups can best use OEP, modify OEP to accommodate each group's unique needs and work flow.
7. **Test OEP**. Test OEP in your development and test environments to ensure that your modifications function as expected.
8. **Install OEP to your production environment**. When you're done modifying and testing OEP, install OEP to your production environment so you can begin the deployment process.
9. **Migrate your modifications**. After installing OEP to your production environment, migrate all desired changes from your development environment to your production environment.
10. **Configure users and user access**. Now that OEP is fully configured in your production environment, set up your users and set their access permissions.
11. **Set default user preferences**. Using a special user account called User Preferences Default, you can establish various preference settings for the OEP system.

12. **Deploy OEP**. You're now ready to deploy OEP to your users.
13. **Administer OEP**. OEP is up and running, and you're now responsible for administering and maintaining it for your users to ensure that their productivity remains high.

What is OEP?

Onyx Employee Portal (OEP) is an enterprise-wide, Web-based software application that enables you to manage all of your customer information from a single user interface.

OEP helps you manage every business interaction with your customers. Using OEP, you can gather, integrate, and share valuable customer information with all members of your organization. You typically enter customer records into OEP as leads or first-time customers, update them during the sales cycle, and maintain them throughout your organization's customer care programs. OEP is fully customizable and can be integrated with other systems.

With this type of insight into integrated customer information, you can maximize both sales effectiveness and service responsiveness. OEP helps you manage activities such as marketing campaigns, sales opportunities, customer service, and technical support, thereby eliminating the problems caused by disparate departmental applications.

The customer-centric design of OEP provides members of your organization with a single resource for the customer data they need to do their job effectively.

Key CRM features

OEP provides full Customer Relationship Management (CRM) capabilities that have traditionally been delivered only through client-server applications. OEP enables you to seamlessly:

- Access a personalized view of customer information
- Access a powerful, integrated database that contains customer, partner, and supplier information
- Manage sales opportunities, support incidents, service requests, contacts, and tasks

OEP's feature set includes the following types of records:

- Customers (companies and individuals)
- Internal and external contacts
- Sales, service, and support incidents and tasks
- Sales forecasts and quotes
- Marketing surveys and campaigns
- Products and literature items
- File attachments (such as to Word documents and any other file that you might want to attach to a customer record
- Comments for saving miscellaneous data about your customers
- Scripting functionality to support routine processes
- Integration with Microsoft Outlook email and appointment calendars

- Integration with the Cognos reporting module
- A set of OEP tools, including:
  - List Manager, which helps you search for customer records so you can perform a certain action on those records via bulk processing
  - Work Tickets and work notes, which enables you to write and track work that may or may not be directly tied to customer records
  - Messenger, which enables you to send messages to other OEP users
  - Notification
  - Navigator
- and much, much more.

For extensive details on these features, see the "OEAS feature overview" in the OEAS Technical Reference.

Key design features

Modifying and extending OEP

- You can fully configure and customize OEP to suit your business needs. Several feature areas of OEP can be modified using UI Configuration Workbench (UCW), which enables you to configure how OEP looks and behaves.
- OEP uses XML to communicate with OEAS in the business services tier.
- OEP uses many non-compiled script files, making the user interface components completely extensible and replaceable.
- OEP contains several ActiveX controls that are downloaded to the client computer to aid in application use.

Deploying and administering OEP

- OEP performs client-side data validation and other local processing, and performs session management using browser cookies.
- OEP offers two types of authentication, allowing you to leverage one or both types within your network environment.

OEP System Architecture

Before you begin to work with OEP, it's important to understand how it fits into the Onyx Internet application framework.

The Onyx Internet application framework is separated into three logical tiers. This tiered architecture (also known as an n-tier architecture) reduces the overhead of round-trip communications with the central application server, as is typical of the traditional two-tier client/server model.

Presentation services. The presentation services tier presents information to, and provides interaction with, the end user. This tier handles all data display, user input, and Web services hosting.

OEP resides in this tier. OEP is an ASP-based application (using both classic ASP and .NET components) running on Microsoft Internet Information Server (IIS). Users access the application through Internet Explorer. OEP uses DHTML, ActiveX controls, XML, and XSL as part of the presentation tier.

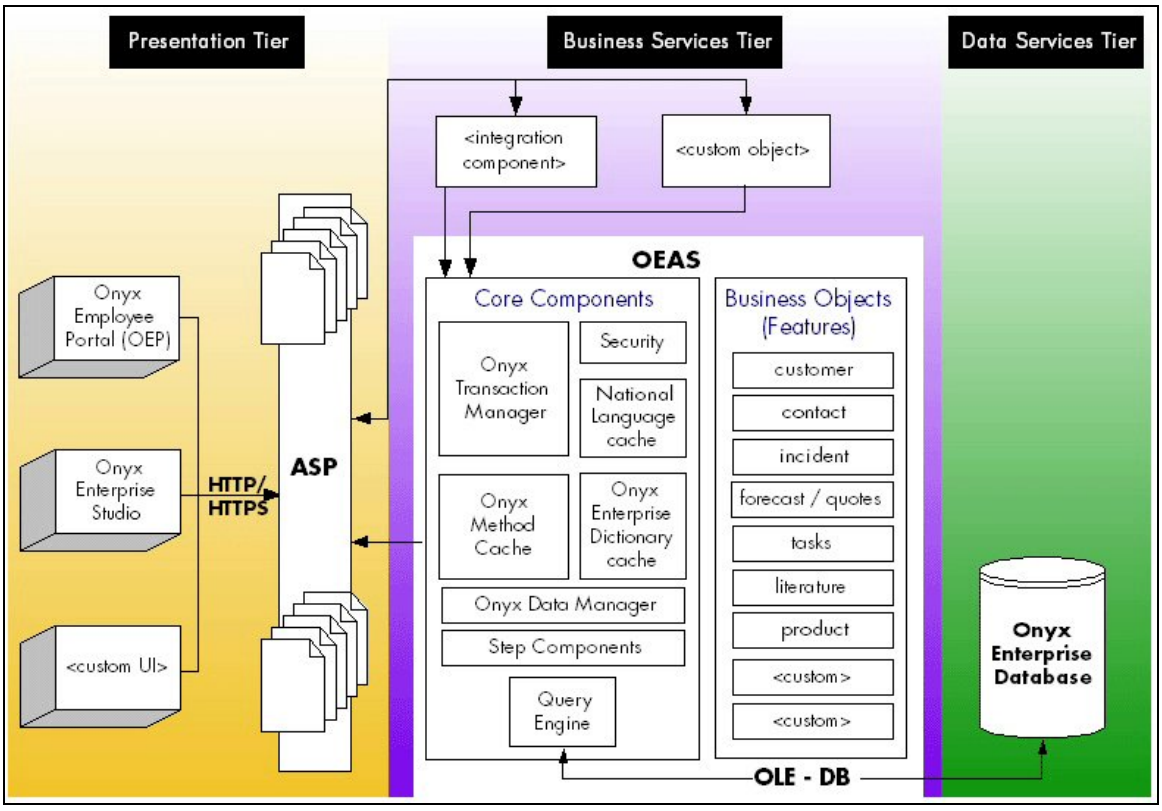
Business services. The business services tier provides reusable application logic that can be used by all presentation services. This includes messaging services, data access, and component services such as transaction management. Data is retrieved from the data store by the business services, which in turn pass the data to the presentation services for interaction with end users. Because the business services tier acts as a broker of information between the presentation services and data services tiers, it is often referred to as the middle tier.

The Onyx Enterprise Application Server (OEAS) is the primary component of the business services tier. OEAS is a Microsoft COM+ application running on Microsoft Windows 2000 or Windows 2003.

Data services. The data services tier provides the ability to define, store, retrieve, and synchronize data.

The Onyx Enterprise Database (OEDB) is the primary component of the data services tier.

The following diagram illustrates how the main Onyx components fit within the n-tier architecture.



Benefits of this architecture

This architecture has the following benefits:

- Ability to scale the business services tier and the data services tier as needed in your corporate environment.
- Minimal client requirements and simplified client deployment.
- Instant client upgrades and quick deployment of customizations.

OEP Profile Architecture

OEP's profile architecture, made possible through UCW, enables you to configure vastly different OEP user interfaces (UIs) using the same installation of OEP. Each profile-specific UI may look and behave entirely differently, or may share as many characteristics as you wish.

UCW enables you to modify the OEP UI according to the needs of each group in your organization, creating distinct UIs that are well aligned with each group's unique work flow. As an OEP administrator, you can grant certain users permission to access only the UI that has been designed for them, and you can grant other users—such as those who work in several different groups—permission to access multiple UIs so that they can log on to OEP using whichever OEP profile will best help them accomplish their task at hand.

For example, let's say you used UCW to configure two different OEP UIs, one for the sales team and another for your marketing team. Using OES, you can assign members of your sales team to the "Sales" OEP profile, and assign members of your marketing team to the "Marketing" OEP profile. You can then assign the manager of both teams to both profiles, so she can access both UIs.

Terminology

The following key terms are essential for understanding the OEP profile architecture:

Term	Definition
OEP UI	The Onyx Employee Portal user interface. You can configure various OEP areas using UCW, and you can customize other areas of OEP by writing custom code.
Profile-specific UI (custom UI)	The OEP UI associated with an OEP profile. The appearance, behavior, and other such characteristics of a profile-specific UI depend on the modifications made to both it and the baseline UI. The default installation of OEP includes one profile-specific UI: the UI associated with the OEP User profile.
Baseline UI	The OEP UI associated with the Global profile. This UI is not visible to OEP users. Configurations made to the baseline UI apply to all profile-specific UIs, whereas configurations made to a profile-specific UI apply to that UI only. UCW includes only one baseline UI: the UI associated with the Global profile. You cannot

Term	Definition
	create alternate baseline UIs.
OEP profile	A security element that provides access to a profile-specific UI. Using OES Security Administration, you can create OEP profiles to define a set of users and/or roles with unique UI workflow requirements. One OEP profile, called OEP User, is provided with the installation of OEP.
Global profile	A security element that provides access to a special UI that is reserved for UCW designers. The UI associated with this profile serves as the baseline configuration for all profile-specific UIs. See "baseline UI," above. This profile is not accessible by OEP users, nor is it listed among the OEP Profiles defined within OES.

Configuring profile-specific UIs

You can configure a custom UI for each set of users/roles that belong to an OEP profile. You can also configure the baseline UI whose configurations apply to all profile-specific UIs.

Within each profile-specific UI, you can configure various OEP areas, including pages such as Company PowerPage, Individual PowerPage, Company, Individual, Incident, Task, and Products. When your OEP users navigate to pages that have been configured via UCW, the configured pages are served to those users based on their OEP profile (that is, the OEP profile they selected when logging on to OEP).

For instance, let's say one of your users belongs to an OEP profile called Sales1. When that user requests the Company PowerPage, the Company PowerPage that you designed specifically for the Sales1 profile will be served to that user, rather than the default non-UCW-configured Company PowerPage.

As discussed in the best practices for configuring OEP, carefully manage your configurations to the baseline UI and each profile-specific UI.

To configure the baseline UI (whose configurations apply to all profile-specific UIs):

1. Log on to OEP (in your development environment). When prompted for a profile, select the Global profile.
2. Configure the desired UCW-enabled areas.
3. If desired, set default user preferences for all UIs.

To configure a profile-specific UI:

1. Using OES Security Administration, create an OEP profile.

   You can assign users and/or roles to OEP profiles at any time.

2. Log on to OEP (in your development environment). When prompted for a profile, select the OEP profile related to the UI you want to design.
3. Configure the desired UCW-enabled areas.
4. If desired, set default user preferences for that UI.

The compile process

It's important to understand how UCW generates (compiles) profile-specific UI pages, because the process influences the way in which you might configure those pages. The key is to remember that modifications to the baseline UI (which is accessed via the Global profile) are inherited by all profile-specific UIs, while modifications made to a profile-specific UI belong to that UI only. (Note that this inheritance model works similarly for user preferences, except that user preferences established for a profile-specific UI override the settings that it had inherited from the Global profile.)

When you compile your changes, UCW merges the changes into their corresponding default OEP pages and generates a set of fully functional ASP pages. During the compile process, UCW first implements all baseline UI modifications (changes made via the Global profile), and then implements all profile-specific UI modifications.

Consider the following example: Let's say that Ed configures the Company PowerPage for the Support and Marketing UIs. Ed wants both UIs to hide the Sales tab, so he makes that change to the baseline UI. (He accesses the baseline UI by selecting the Global profile when he logs on to OEP.) He then compiles his changes and logs out of OEP. Ed also wants to add a "Support" caption to the Support UI, and a "Marketing" caption to the Marketing UI, so he makes those changes to the two profile-specific UIs. He first makes his change to the Support UI, compiles his changes, and logs out of OEP. He then makes his change to the Marketing UI, compiles his changes, and logs out of OEP.

Example

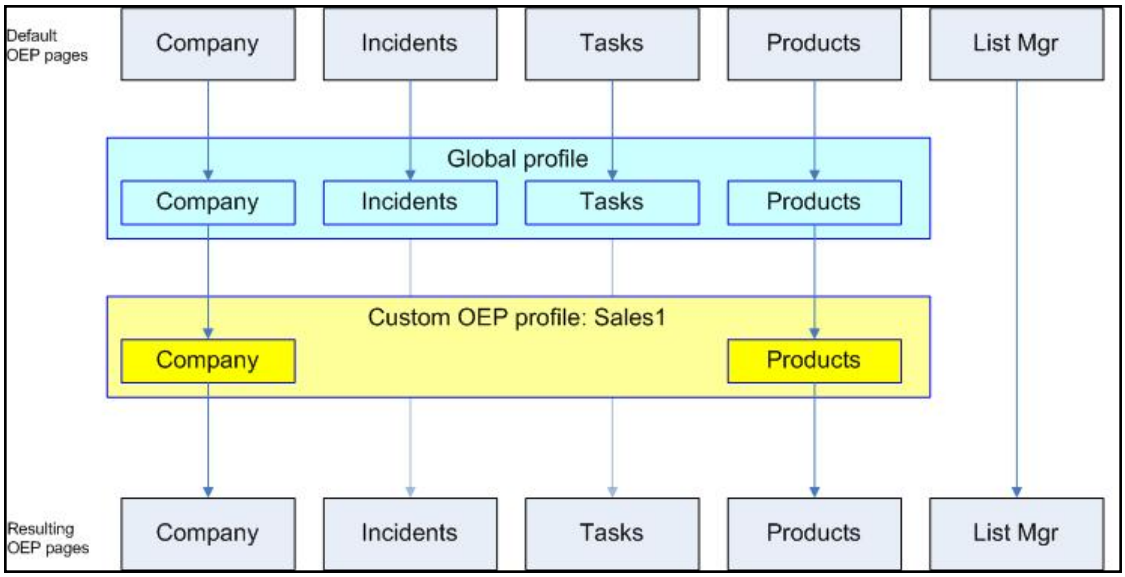
The following graphic illustrates the results of configuring the Sales1 UI using the Global profile and a sample OEP profile called Sales1. In this example, the UCW designer has configured four OEP pages using the Global profile; these changes are inherited by the Sales1 UI (and all other profile-specific UIs). The UCW designer has also configured two of those OEP pages—the Company and Products pages—using the OEP profile called Sales1; these modifications apply only to the Sales1 UI.

The compile process first applies all changes made to the baseline UI (accessed via the Global profile), and then applies all changes made to Sales1 UI (associated with the custom OEP profile called Sales1). After the pages are migrated to the production environment, the resulting OEP pages are presented to all OEP users who have logged on to OEP using the Sales1 OEP profile.

In this example, when your OEP users access the Company and Products pages, the pages presented to them contain changes made to the baseline UI and to the Sales1 UI. When they access the Incidents and Tasks pages, on the other hand, the pages presented to them contain changes

made to the baseline UI only. The compile process does not impact the List Manager and Task Manager pages, because those pages are not UCW-enabled and therefore cannot be modified by UCW.

Note: This graphic serves only as an abstraction of the compile process, as seen from an OEP user's perspective. It does not identify all UCW-enabled areas, nor does it attempt to explain the details of the compile process.



Global Profile

The Global profile is a security element that provides UCW designers access to the baseline UI. Only one baseline UI is provided with the installation of OEP.

Configurations made to the baseline UI apply to all profile-specific UIs (including the OEP User UI provided with the installation of OEP), whereas configurations made to a profile-specific UI apply to that UI only. The baseline UI is not visible to OEP users.

Let's say, for instance, that you have created several different OEP profiles and that you want to make the same modification to all profile-specific UIs. You can accomplish this task by making the change just once—in the baseline UI (accessed with the Global profile)—since all changes to the baseline UI are automatically applied to all profile-specific UIs. As we recommend in our best practices, it's usually best to configure the baseline UI first, and then configure your various profile-specific UIs

When configuring OEP for the various teams in your organization, determine whether you want the configuration to apply to all profile-specific UIs, or to just one or several profile-specific UIs. If you want the configuration to apply to all profile-specific UIs, log on to OEP using the Global profile and make your changes to the baseline UI. To configure a profile-specific UI, log on to OEP using the related OEP profile.

The Global profile is available to UCW designers and other users who have been granted access to either the OEP.administrator role or the UI:OEP.ucw.configure resource. The Global profile is not accessible by OEP users, nor is it listed among the OEP Profiles defined within OES Security Administration. The Global profile is not generally accessible in the production environment, although UCW designers with appropriate access privileges can log on to a production installation of OEP to set default OEP user preferences.

OEP User Profile

The OEP User profile is a security element that provides access to the OEP User UI, a profile-specific UI. The OEP User UI is the only profile-specific UI provided with the installation of OEP.

If you upgraded from an earlier version of OEP, your users may have automatically been assigned to this profile (users who belong to OEP.Administrator, OEP.PowerUser, and OEP.User roles are automatically assigned to the OEP User profile). If you plan to configure custom UIs for several teams in your organization, however, you must create additional OEP profiles and then assign your users to these new OEP profiles as desired.

Like all custom UIs, the appearance, behavior, and other such characteristics of the OEP User UI depend on the modifications made to both it and the baseline UI.

In the default installation of OEP, the OEP User UI has the same configuration as the baseline UI: their individual edit page, incident edit pages, Header Bars, and so on, are identical. Configure the OEP User UI as desired to meet your company's needs, or create your own OEP profiles and configure them as desired.

The OEP User profile is available to users who have been granted access to any of the following OEP roles:

- OEP.Administrator
- OEP.PowerUser
- OEP.user

The OEP User profile is listed among the OEP Profiles defined within OES Security Administration.

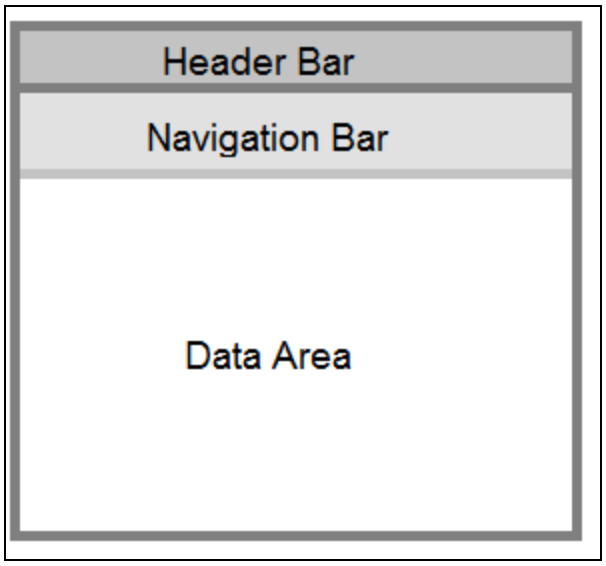
None Profile

The OEP profile called None is a security element that supports the Global profile.

Warning: Do not activate or modify the None profile.

OEP Frame Layout

The following illustration depicts the basic frame layout for the OEP user interface.



These frames host the following types of information:

Header Bar. Contains buttons from which you can navigate to the most commonly used features of OEP.

QUICK! Tools. Contains the QUICK! search and QUICK! list tools that enable you to find customer records.

Navigation Bar. Contains a hierarchical list of the primary OEP features.

Data Area. Displays the main application forms, such as the PowerPage, Task Manager, and Products. This frame is sometimes subdivided into multiple frames.

A few features, such as List Manager, Incidents, and Messenger, appear in their own windows and not in the Data Area frame.

2

OEP Configuration

Configure or Customize OEP?

You know that you want to modify OEP, but should you configure it, or customize it? What's the difference?

Configuring OEP. Essentially, configuration tasks are those that can be accomplished using an Onyx tool such as UCW, OES, or any of the many other Onyx tools and utilities. Some configuration tasks require you to modify existing code or write some of code of your own, but in cases like these, the code writing can be done exclusively within a tool (as is the case with writing custom JavaScript within the UCW Dynamic Forms Designer). For details, see the Configuring OEP book.

Customizing OEP. Customization tasks, on the other hand, are those that cannot be accomplished using an Onyx tool and must therefore be done by writing your own custom code. Customizing OEP requires an understanding of the OEP programmer's reference, just as customizing OEAS requires knowledge of the OEAS reference libraries (which are documented in the OEAS Technical Reference). For details, see the Customizing OEP book.

Still, it may be unclear in certain circumstances whether you should configure OEP using UCW, or whether you should customize OEP by writing custom code. Consider the following guidelines when making your decision.

Configure OEP...

- ...if you plan to modify a UCW-enabled area of OEP
- ...if you want to modify how users navigate through OEP (using the Header Bar and Navigation Bar)
- ...if you want to modify the appearance of OEP

Customize OEP ...

- ...if you plan to modify an area of OEP that is not UCW-enabled
- ...if you want to create a new OEP page
- ...if you want to modify an OEP page so dramatically that it deserves to be a new, custom page

Configuring OEP

You can modify certain areas of OEP using UI Configuration Workbench (UCW). Benefits of UCW configurations include:

- **Ease of configuration**: UCW provides WYSIWYG configuration of many areas in OEP, enabling you to configure, compile, and test your configurations, including changes to user navigation and form presentation and behavior.
- **Upgradeability**: When you configure OEP using UCW, you ensure that your UI configurations will be upgraded to the next version of OEP without requiring you to rewrite or re-implement your modifications. Customizations that are implemented outside of UCW, however, are not automatically applied when OEP is upgraded.
- **Custom OEP pages for different teams**: Using UCW and OEP Profiles, you can configure the same OEP page differently for different teams in your organization, enabling profile-based user navigation and page presentation and behavior. You can apply your UCW configurations to one or more custom UIs (accessed via the related OEP profiles) or to all custom UIs at once (using the baseline UI accessed with the Global profile).

Using UCW, you can simplify navigation to make workflow more intuitive and data gathering more efficient. For example, you can change page layout, configure Header Bar buttons, add custom UI controls to pages, remove existing UI controls from pages, and modify UI text (captions and labels) within pages. To understand the UCW configuration workflow, see Getting Started. For detailed use cases of UCW configurations, see Sample UI configurations.

Tip: While many UCW configurations require little more than Onyx product knowledge, some tasks require programming experience. When assigning configuration tasks, ensure that the individual's skill set meets the task requirements.

UCW-enabled areas

Several areas within OEP are UCW-enabled. The function, behavior, and appearance of these areas are profile-specific. OEP areas not listed in this section are not UCW-enabled.

Note: The User Preferences set by OEP users are profile-specific; that is, an OEP user can set unique User Preferences for each OEP profile he or she belongs to.

Header Bar

You can add, modify, hide, and delete OEP's Header Bar buttons. For more information, see Configuring the Header Bar.

Navigation Bar

You can add, modify, hide, and delete OEP's Navigation Bar links and link groups. For more information, see Configuring the Navigation Bar.

Pages

You can add, modify, hide, and delete UI controls and other page elements by OEP profile on the following UCW-enabled pages. Default tabs on these pages are UCW-enabled unless otherwise noted. Custom tabs that you create are always UCW-enabled. UCW configurations on incident, task, and product pages apply to the selected category only.

For more information on UCW configuration of page elements, see About design mode.

UCW-enabled page	Website folders for related files (Do not customize; see Best practices for configuring OEP)
Company PowerPage (Details tab only)	\powerpage
Individual PowerPage (Details tab only)	\powerpage
Company edit page	\customer
Individual edit page	\customer
Incident edit page (toolbar on the Notes tab only; configurations are category-specific)	\incident
Task edit page (toolbar on the Notes tab only; configurations are category-specific)	\incident
Products page	\Products
Product edit page (toolbar on the Product Line tab only)	\Products
Product line edit page	\Products

Limited UCW configuration is available for intrinsic (default) controls that contain multiple entries, such as Telephone and Billing Address on the company PowerPage.

Important: Data-binding is not available for toolbox controls added to custom tabs on the Products page (product summary page) because this page is based on the retrieveList method. The methods retrieveList and retrieveCollection are not supported for data-binding in UCW.

Items not UCW-enabled on these pages include default toolbar titles, result list controls (RLCs), search areas, and separately launched windows, such as the Other Addresses window launched from the company edit page. RLCs can have profile-specific display preferences, such as column widths.

Custom code is required to modify OEP areas that are not UCW-enabled. For more information, see Customizing OEP.

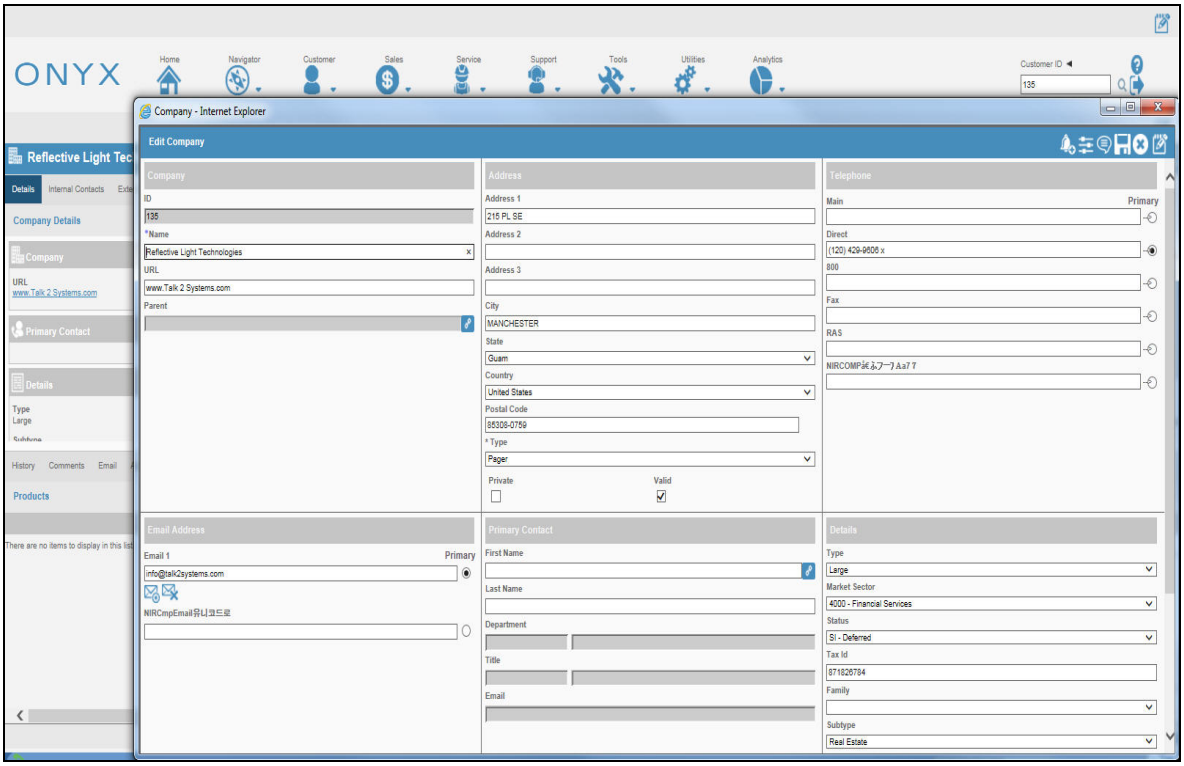
Note: Pages that are transmitted using OEP's print and email buttons (and) do not contain UCW configurations. To print the page as it appears on the screen, use the browser's print button. (For more information, see *Printing and emailing UCW-configured pages*.)

Configurable page elements

You can add, modify, hide, and delete UI elements on the Header Bar, Navigation Bar, and any UCW-enabled page. For example, you can add and remove Header Bar buttons, Navigation Bar links, tabs, and UI controls and you can configure toolbars, tabs, page behavior and appearance, UI text, and messages.

The following graphic illustrates the company edit page in front of the company PowerPage. To learn about a specific page element in this example graphic and to see if the page element is configurable using UCW, point to the element; click to see related configuration information.

Note: See also the illustration of a UCW-enabled page during design mode.



Accessing UCW

You can access UCW (that is, enter design mode) from any UCW-enabled page, as well as from the main application frame. UCW access is available in the development environment only; a properly configured user account is required. For more information, see Configure your development environment.

When working in design mode, you can access various UCW tools to configure OEP.

Entering design mode

You can enter design mode from any UCW-enabled page and from the main application frame.

To enter design mode for:	Do this:
A page	Click on the desired page.
The main application frame	Click above the Header Bar.

Exiting design mode

You can exit design mode for the current context (page or main application frame), profile, and category (for incidents), by doing any of the following:

- Saving configurations
- Canceling configurations from the current UCW session
- Resetting all configurations (cancels configurations from all UCW sessions)

> **Note:** When you reset configurations for a profile-specific UI, the UI becomes identical to the baseline UI (as modified with the Global profile) for the selected context (page or main application frame) and incident category, if applicable.

To save configurations and exit design mode:

- Click on the UCW toolbar.

To cancel configurations from this session and exit design mode:

In design mode for this context...	Do this...
Main application frame	Click on the UCW toolbar.
Page that is displayed in the Data Area frame	Click on the UCW toolbar.
Page that is displayed in a separate browser window	Close the browser window.

To reset all configurations and exit design mode:

- Click  on the UCW toolbar.

example of design mode for the main application frame

About Design Mode

When you enter design mode, the UCW toolbar appears on the upper right of the page (or the upper right of the Header Bar, when entering design mode for the main application frame).

When you enter design mode for a selected page, the page changes appearance to indicate page elements such as sections and UI controls.

What would you like to do?

- Learn about the UCW toolbar
- Learn about page elements
- Learn about color-coded design elements
- View an example of page elements
- View an example of design mode for the main application frame

The UCW toolbar

The UCW toolbar, which displays when you enter design mode, includes the following buttons.

Button	Tooltip	Description
	Hide Controls Pane	Hides Intrinsic Controls, Toolbox Controls, and Properties, removing them from the window and increasing the area of the window occupied by the page.
	Dock Controls Pane	Displays Intrinsic Controls, Toolbox Controls, and Properties on the left of the window, decreasing the area of the window occupied by the page.
	Show Hidden UI Elements	Displays all UI elements currently hidden using configured page behavior (action statements configured in Dynamic Forms Designer).
	Show Intrinsic Controls Pane	Hides or displays Intrinsic Controls (available when docking is disabled only).
	Show Toolbox Controls Pane	Hides or displays Toolbox Controls (available when docking is disabled only).
	Open Navigation Designer	Displays Navigation Designer for configuration of the Header Bar, Navigation Bar, toolbars, and tabs. For more information, see Configuring page navigation.
	Open UI Text Editor	Displays UI Text Editor for configuration of UI text and messages. For more information, see Configuring UI and message text.
	Open Dynamic Forms Designer	Displays Dynamic Forms Designer for configuration of page behavior and appearance. For more information, see Configuring page behavior and

Button	Tooltip	Description
		Configuring page appearance (ASP and HTML).
	Reset Page Configurations	Discards all configurations for the selected page and profile and exits UCW, ending the UCW session.
	Save page configurations and exit design mode	Saves the current session's configurations for the selected page and profile and exits UCW, ending the UCW session.
	Compile	Generates a set of ASP pages from your UI modifications for the current profile. For more information, see Compiling your changes.
	Exit Design Mode	Discards the current session's configurations for the selected page and profile and exits UCW, ending the UCW session.  **Note:** This button is displayed on the main application frame, the PowerPage, and the Products page (product summary page) only. To discard configurations from the current session on other pages, close the page's browser window.

Design mode page elements

When you enter design mode for a page, you can add, modify, and delete UI controls and other page elements. The Intrinsic Controls, Toolbox Controls, and Properties panes automatically appear on the left of the page. The Intrinsic Controls pane lists the intrinsic (default) controls that are removed from the page. The Toolbox Controls pane lists the types of toolbox controls you can add to the page. The Properties pane includes definitions for the selected control.

Color-coded design elements

Color-coded design elements include canvases, panels, sections, and UI controls (toolbox controls and intrinsic controls). A canvas can contain multiple panels, which can each contain multiple sections. (A panel can contain multiple canvases, as well.) Controls can be added to panels and sections. The border and corner of each canvas, panel, and section is color-coded, as well as each control's selector.

The following color-coded design elements are displayed during design mode:

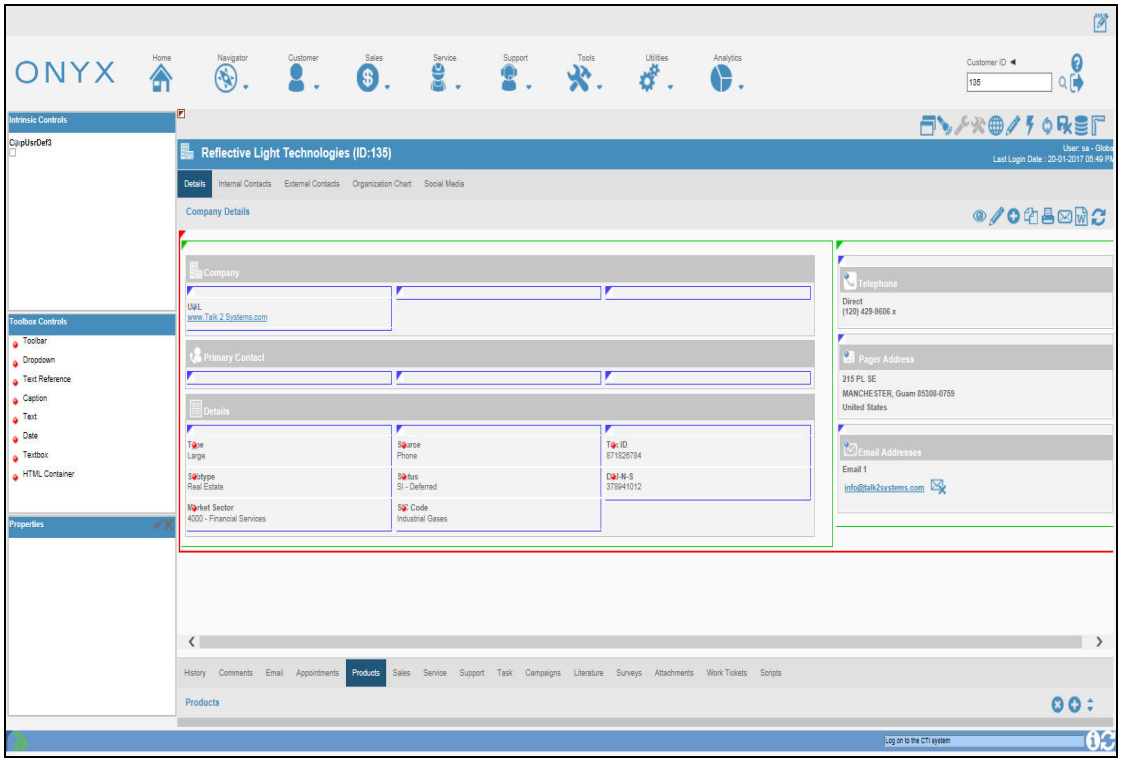
Design element	Description	Function
	Canvas corner (red triangle) located on the upper left of the canvas.	Displays a shortcut menu when right-clicked.
	Panel corner (green triangle) located on the upper left of each panel in the canvas.	Displays a shortcut menu when right-clicked.

Design element	Description	Function
	Section corner (blue triangle) located on the upper left of each section in a panel.	• Displays a shortcut menu when right-clicked. • Relocates the section when dragged.
	Control handle (red circle) located on the left of each toolbox control.	• Displays a shortcut menu when right-clicked. • Relocates the control when dragged.
	Control handle (blue circle) located on the left of each intrinsic control.	• Displays a shortcut menu when right-clicked. • Relocates the control when dragged.

See also Configuring layout and Configuring UI controls.

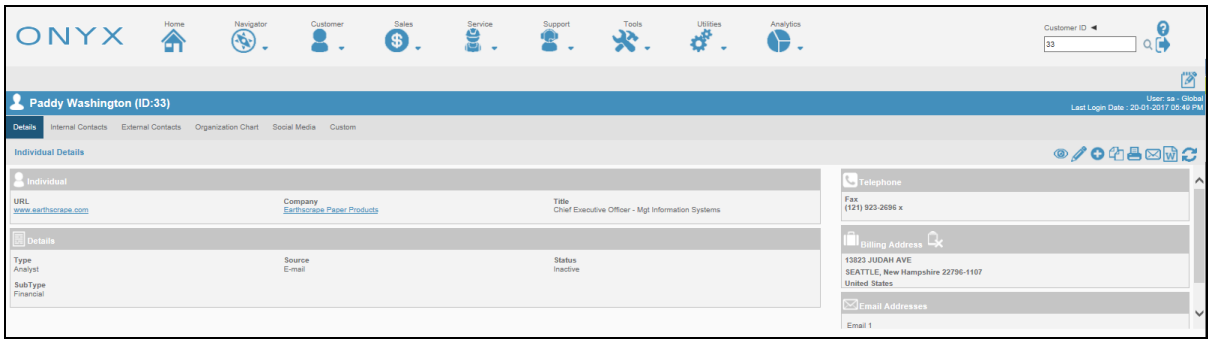
Example of design mode page elements

The following graphic illustrates design mode for the company edit page. This example graphic includes custom tabs and other page elements, such as UI controls and captions. Color-coded design elements, such as red control handles, are also included. To learn about a specific item in this example graphic, point to the item; click to see related configuration information.



Example of design mode for the main application frame

The following graphic illustrates design mode for the main application frame. This example graphic includes a custom Header Bar button ("Analytics"); the Header Bar has been configured to hide the List Manager and Search buttons for this profile. To learn about a specific item in this example graphic, point to the item; click to see related configuration information.



Using UCW Tools

You can use UCW tools to configure UI text and page navigation, behavior, and appearance.

UCW provides the following tools for modifying the OEP UI. To access these tools, you must first enter design mode.

UCW tool	Description	To launch this tool:
Navigation Designer	Enables you to modify how users navigate from one OEP page to another by configuring the following areas: • Header Bar • Navigation Bar • Toolbars • Tabs	Click on the UCW toolbar.
UI Text Editor	Enables you to modify UI text and messages.	Click on the UCW toolbar.
Dynamic Forms Design	Enables you to modify page behavior (responses to events) and page appearance (custom files and style sheets).	Click on the UCW toolbar.
Canvas Designer	Enables you to modify page layout and TAB key order for the panels.	Enter design mode for a UCW-enabled page, right-click the upper left corner of a panel ( ), and then select Convert to Canvas. **Note:** Configurations saved in Canvas Designer replace

UCW tool	Description	To launch this tool:
		all existing items within the panel.

Note: For information on modifying the UI using design mode elements (without UCW tools), see About design mode, About Intrinsic Controls, About Toolbox Controls, and Control properties reference.

Best Practices for Configuring OEP

Consider the following recommendations when configuring OEP.

Don't:

Don't customize UCW-enabled areas

Use UCW to configure UCW-enabled OEP areas. Don't write custom code directly to these areas. Customizing UCW-enabled areas complicates the process of upgrading from this version of OEP to the next.

Don't modify source files for UCW-enabled areas

Don't modify the source files of UCW-enabled OEP areas, and don't directly modify any file that is stored within the UCF directory (other than to the ucf/data/custom/directory, which is intended for your use). Modifying source files complicates the process of upgrading from this version of OEP to the next.

For broad configurations, consider implementing the HTML Container (using UCW) or creating an entirely custom OEP page (which you cannot do using UCW).

Do:

Use UCW tools wherever possible

UCW provides several tools that enable you to configure OEP in ways that previously could be accomplished only by writing custom code. UI configurations that you implement with UCW can be upgraded from one version of OEP to the next without requiring you to re-implement them.

Coordinate with other designers

UCW supports a multi-user development environment, meaning that several designers can configure OEP at the same time. It is important that you coordinate your configuration tasks with those of the other designers.

Ensure that:

- Only one UCW designer works on a UCW-enabled area at one time, per OEP profile. If multiple designers configure the same area of the baseline UI or profile-specific UI simultaneously, unpredictable results may occur.

- You compile all changes to your custom UI before publishing it to your test or production environments.

Test your changes frequently

It's possible to design a page that may not render as expected. Using UCW, for instance, it is possible to add such lengthy captions that their full text cannot be read, and it's possible to add a toolbar with so many buttons that not all of them can fit on the page (and therefore cannot be clicked by users).

Therefore, test each of your modifications frequently to ensure that they look and behave as expected.

Back up your work

We strongly encourage you to back up your configurations at various stages of development. Save and back up your work frequently.

Before making substantial modifications to your OEP configuration, back up all UCW files located in your YourOEPwebsite\ucf\data directory.

Use source control to manage your configurations

Using source control provides you a much finer degree of history and change control than machine-level backup alone.

Place custom files in the recommended folders

Place custom files in the following folder: YourOEPwebsite\ucf\data\custom\company_name

Recommended Approach

As suggested in the OEP Installation Guide, we recommend that use three separate environments for configuring, testing, and using OEP. By keeping the environments separate, you can manage your configurations more easily and better ensure the integrity of each distinct environment.

Development environment	Installing to your development environment gives you access to UCW, which includes the tools you need to modify many aspects of the OEP UI. The development environment requires OEP, OES, OEAS, OEDB and optionally, OPS. Development must be isolated from the test and production environments. As you plan your server configuration, keep in mind that the configurations made in the development environment must be replicated to the test and production environments.
Test environment	The test environment also requires OEP, OES, OEAS, OEDB and optionally, OPS. Your test environment is installed as a Production-type installation, and its server must be configured to match that of your production environment. Migrate all OEP configurations from your development environment to this environment, and test them here. If you need to modify your configurations, implement the changes in your development environment, migrate those changes to your test environment, and test the changes as appropriate. Repeat this process for each series of changes that you make.
Production environment	The production environment requires OEP, OES, OEAS and OEDB on at least one server and, optionally, OPS on its own separate server. After you're done configuring

	and testing each custom OEP UI that you create, migrate all OEP configurations from your development environment to this environment.

Onyx recommends the following approach to configuring OEP:

1. In your development environment, implement common (shared) modifications: Log on to OEP using the Global profile, and then use UCW to implement modifications that you want to apply to all of your profile-specific UIs.
   - Configure page layout, navigation, and UI text. Test your changes.
   - Configure page behavior and UI messages. Test your changes.
   - Configure page appearance. Test your changes.
2. In your development environment, implement profile-specific modifications: For each custom UI, log on to OEP using the desired OEP profile, and then use UCW to modify that UI.
   - Configure page layout, navigation, and UI text. Test your changes.
   - Configure page behavior and UI messages. Test your changes.
   - Configure page appearance. Test your changes.
3. Compile your changes.
4. Migrate your changes from your development environment to your test environment.
5. In your test environment, test the changes you made to each custom UI.
6. From your development environment, migrate your changes to your production environment.
7. Perform some high-level testing in your OEP production environment before deploying OEP to your users.

Tips & Tricks

Consider the following tips and tricks for a successful UI configuration.

Plan your implementation

Plan your OEP implementation carefully:

- Create mockups of the UI, including each OEP page that you plan to modify, and have your users test them to ensure that the design satisfies their needs.
- Determine how many custom UIs you think you will need to create. You will likely want to create an OEP profile for each set of users who use OEP to accomplish tasks using a distinctly different work flow.
- Note that it can be labor intensive to recreate a page if you want to make elaborate changes to its layout, because you may need to delete the page's main canvas and completely rebuild it.

Configure the baseline UI first

After you understand how you want to configure each of your custom UIs, determine which configurations belong in the baseline UI (the UI associated with the Global profile) and which configurations belong in each of your profile-specific UIs. Remember that all configurations implemented in the baseline UI (which is accessed by the Global profile when logging on to OEP) are inherited by all profile-specific UIs (which are accessed by their corresponding OEP profile when logging on to OEP). Modify the baseline UI only when you are certain that you want the modification to apply to all other custom UIs, as well.

You can configure the baseline UI at any time, regardless of how many custom UIs you are currently configuring or have already published to your test or production environments. Remember, though, that any modifications you make to the baseline UI may impact the configurations you have made to each profile-specific UI.

As always, thoroughly test all of your configurations.

If you modify the baseline UI after configuring profile-specific UIs...

If you configure or reset the baseline UI (the UI associated with the Global profile) after configuring a profile-specific UI, be sure to test both the baseline UI and all profile-specific UIs to ensure that everything functions as expected. Because all profile-specific UIs inherit configurations from the baseline UI, changing or resetting page configurations in the baseline UI may produce unpredictable results with corresponding pages in the profile-specific UIs.

For instance, hiding, removing, or otherwise modifying the baseline UI's canvases, panels, sections, and UI controls can greatly affect your profile-specific UIs. Modifying action statements or otherwise modifying the behavior of the baseline UI can affect your profile-specific UIs in a more subtle (or equally obvious) fashion.

If your modifications to the baseline UI caused problems in your profile-specific UIs, you may need to troubleshoot your changes by disabling custom actions and resetting your configurations.

Consider hiding (rather than deleting) UI elements in the baseline UI

If you are considering deleting a UI element in the baseline UI, consider hiding it instead—particularly if you have already configured any profile-specific UIs. Hiding UI elements generally has fewer repercussions on profile-specific UIs than deleting them.

Test your pages after hiding, removing, deleting, or disabling toolbox controls and intrinsic controls

Before you hide, remove, delete, or disable either a toolbox control or an intrinsic control, consider which action statements depend upon that control in order to function. Then use Dynamic Forms Designer to remove, disable, or modify those action statements, and test your pages as appropriate.

Toolbox controls can be hidden, disabled, or deleted. If you delete a toolbox control, action statements created in standard view of Dynamic Forms Designer are automatically deleted, as well. Custom action statements created in Advanced View of Dynamic Forms Designer, however, must be removed, disabled, or modified manually.

If you hide, remove, or disable a toolbox control in the baseline UI, you must also modify all action statements that reference that control in all profile-specific UIs.

Intrinsic controls can be hidden, disabled, or removed from the page. Removing an intrinsic control from the page does not have any effect on action statements created in Dynamic Forms Designer. Therefore, if you remove an intrinsic control from the page, you must remove, disable, or modify any action statements associated with that control.

If you hide, remove, or disable an intrinsic control in the baseline UI, you must also modify all action statements that reference that control in all profile-specific UIs.

Configuring Layout

You can configure the page layout of any UCW-enabled OEP page, as well as any new tab that you add to a UCW-enabled page using canvases, panels, and sections. A canvas defines the structure, or layout, of a page. Using Canvas Designer, you can configure the layout of any UCW-enabled page.

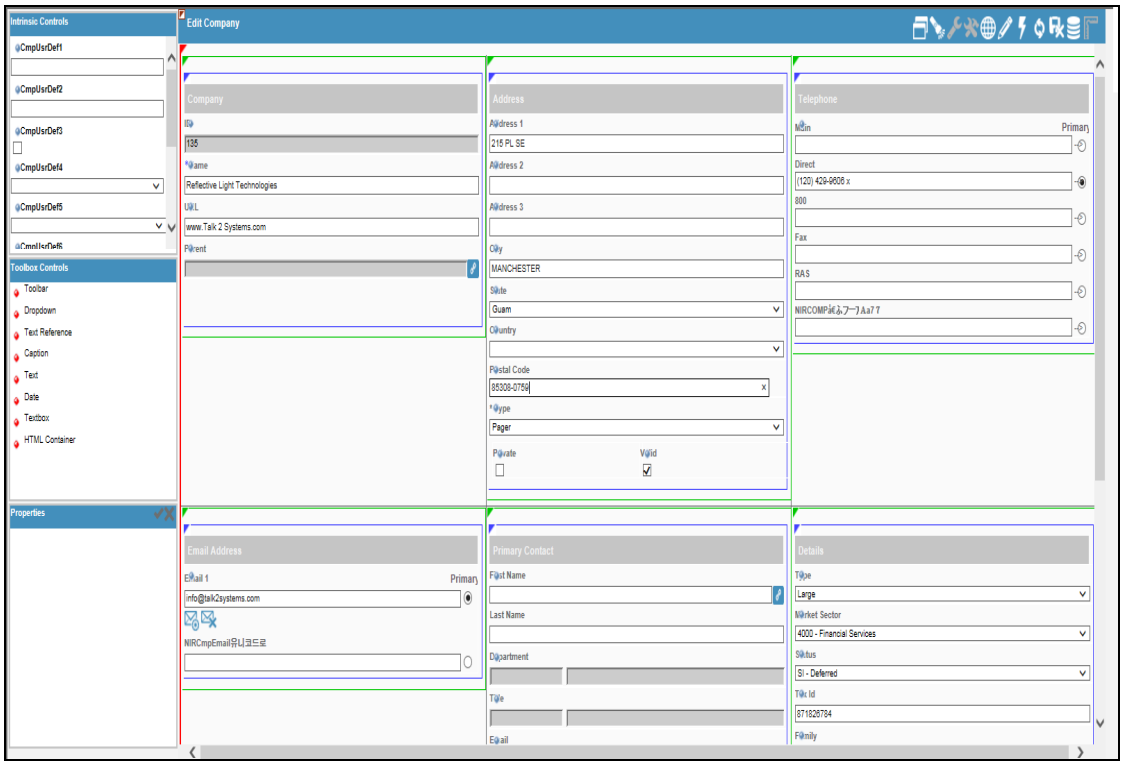
It may be helpful to think of a canvas as a table that consists of columns, rows, and cells. In the UCW environment, however, cells are referred to as panels. A canvas can have any number of panels, and each panel can contain any number of sections. You can also convert a panel into a canvas, which in turn can contain additional panels and sections. Because there is no limit to the number of canvases, panels, and sections that you may add to a page, you will want to put your UI design skills to work to ensure that you're creating a page that your users will find to be both useful and intuitive.

When you enter design mode from any UCW-enabled page, various design elements indicate how the page has been designed. Each UI control is identified by a marker of its own, and each canvas, panel, and section is framed in a different color:

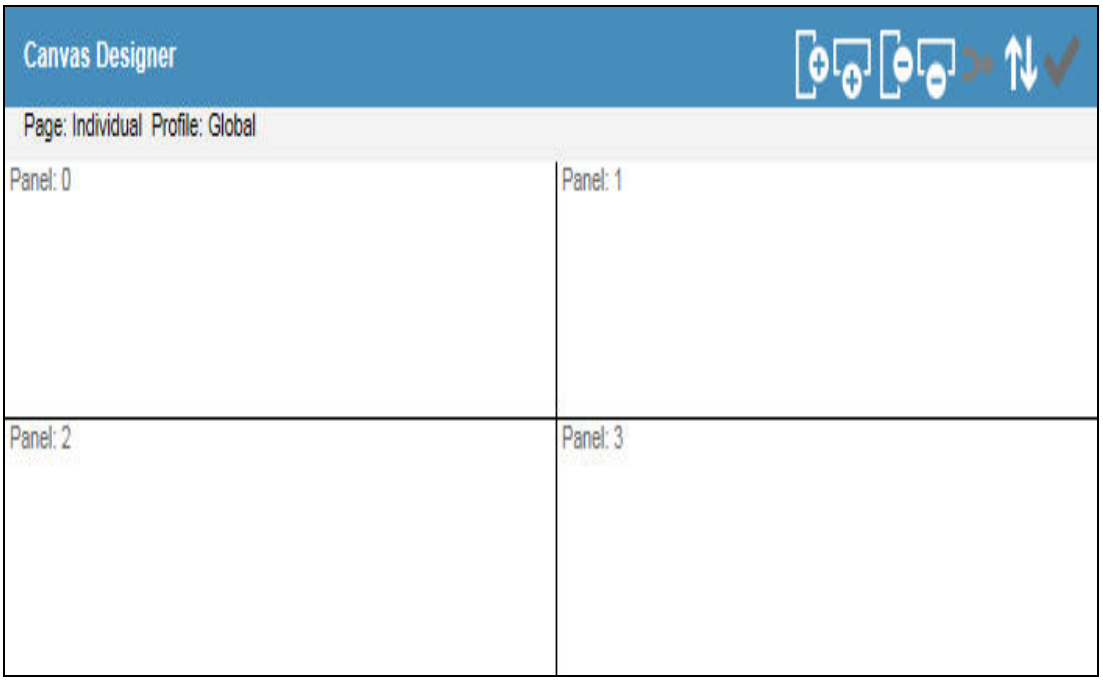
- **Canvases** are framed in red. The top-left corner of each canvas is marked by a red triangle ( ) that, when right-clicked, displays a list of actions that you can perform on that canvas.
- **Panels** are framed in green. The top-left corner of each panel is marked by a green triangle ( ) that, when right-clicked, displays a list of actions that you can perform on that panel. It also serves as a drop-spot for UI controls, captions, or sections that are dragged to it.
- **Sections** are framed in blue. The top-left corner of each section is marked by a blue triangle ( ) that, when right-clicked, displays a list of actions that you can perform on that section. It also serves as a drop-spot for UI controls or captions that are dragged to it, and as a drag handle that enables you to drag the section to another location within the panel or to another panel.

Sample layout

The following graphic shows a sample Edit Company page in design mode.



If you were to use Canvas Designer to create a page with this same structure, your canvas would look like this:



Setting Page-level Properties

You can set two page-level properties for each UCW-enabled page: Minimum Width and Label Width.

Setting the width of a page

Using the page's Minimum Width property, you can set the default width of a page. When users resize the page to be smaller than this default width, a horizontal scroll bar appears.

To specify the width of a page:

1. Right-click the page properties image () located in the upper-left corner of the page, and select **Properties**.
2. In the **Minimum Width box**, specify the pixel width for the page.
3. Click  to apply changes.
4. Close the Properties pane.

Setting the width of control labels

Using the page's Label Width property, you can set the default label width for all UI controls' labels whose label width isn't specified within a panel. You can also set the label width for all UI controls within a panel.

Label width determines the amount of space reserved for a UI control's label. If a control's label exceeds the specified width, the text wraps.

To specify the default width of control labels:

1. Right-click the page properties image () located in the upper-left corner of the page, and select **Properties**.
2. In the **Label Width box**, specify the default pixel width for labels.
3. Click  to apply changes.
4. Close the Properties pane.

Showing Hidden UI Elements

You can show UI controls (including captions), control labels, tabs, and sections that were dynamically hidden using configured page behavior (action statements configured in Dynamic Forms Designer). This function is helpful when you want to ensure that a UI element was not already created, or when you want to modify a dynamically hidden UI element. Dynamically hidden control toolbar items (such as buttons) are not shown with this function.

When you show hidden UI elements, they are displayed on the page.

On the PowerPage, controls with no data ("empty controls") are automatically hidden; these controls are shown along with dynamically hidden UI elements when you show hidden UI elements. For example, if the PowerPage does not display the intrinsic "URL" control because the current record has no defined URL, you can show this "URL" control.

Note: When you show hidden UI elements, they remain shown until you end the current UCW session.

To show all hidden UI elements:

- Click  on the UCW toolbar.

Working with Canvases

Canvases are framed in red. The top-left corner of each canvas is marked by a red triangle (◤) that, when right-clicked, displays a list of actions that you can perform on that canvas.

What would you like to do?

- Create a canvas (convert a panel into canvas)
- Remove a canvas (convert the canvas to a panel)

Procedures

Create a canvas (convert a panel into a canvas)

Create a canvas to design the layout of the page (or certain area of the page). You can create a canvas from any panel of a UCW-enabled page or tab. To create a canvas, you convert an existing panel into a canvas; you can't create a canvas from scratch.

Tip: Converting a panel into a canvas removes all UI controls from that panel. Because data binding for toolbox controls is removed as well, you may want to temporarily relocate the controls to another panel to save them for future use.

To convert a panel into a canvas:

1. Locate the panel that you want to convert to a canvas.
2. If the panel contains toolbox controls that you may want to use later, drag the controls to another panel.
3. Right-click the panel's green triangle ( ) and select Convert to Canvas.
4. Using Canvas Designer, design the canvas as desired.

Remove a canvas (convert the canvas to a panel)

You can't exactly remove a canvas from a page, but you can convert the canvas into a panel. When you change a canvas into a panel, all configurations for that canvas are lost.

To convert to panel:

1. Locate the canvas that you want to convert to a panel.
2. If the canvas contains toolbox controls that you may want to use later, drag the controls to another canvas.
3. Right-click the canvas's red triangle ( ), and select Convert to Panel.

Using Canvas Designer

Canvas Designer enables you to define the layout of a page, or of a certain area of the page. Canvases are comprised of one or more panels organized into columns and rows. Using Canvas Designer, you can:

- Specify the number of panels in the canvas
- Organize the panels into any number of rows and columns
- Merge multiple panels into a single panel
- Assign the TAB order for the canvas

To access Canvas Designer, right-click a panel's green triangle ( ) and select Convert to Canvas.

What would you like to do?

- Design a canvas (from start to finish)
- Add panels
- Remove panels
- Merge panels
- Set TAB order
- Apply the canvas

Procedures

Design a canvas

The design of a page's layout is often driven by users' work flow requirements, which you may have already assessed when planning your OEP implementation. Because canvases serve as the foundation of page layout, it's important to design them with your users' needs in mind.

Canvases cannot be edited after they are saved, so design your canvases carefully. If you need to re-design a canvas, you must convert the canvas to a panel, and then re-create the canvas by converting the panel to a canvas.

To design a canvas:

1. Access Canvas Designer by right-clicking a panel's green triangle ( ) and selecting **Convert to Canvas**.
2. Use Canvas Designer, add or remove panels from the default canvas.
3. Merge one or more panels.
4. Ensure that the canvas is designed as desired. After you enter TAB order mode, you cannot add, remove, or merge panels.
5. Set TAB order for the canvas.
6. Apply the canvas to the page.

Add panels

You can add any number of panels to a canvas. You can then merge panels to further enhance page design.

Adding panels to a canvas resets all changes that you have made to the canvas.

To add panels to a canvas:

- Click  to add a row of panels.

  OR

- Click  to add a column of panels.

Remove panels

You can remove rows and columns of panels from a canvas. You cannot, however, remove a single panel from the middle of a row or column (instead, you can merge the panel with another panel).

Removing panels from a canvas resets all changes that you have made to the canvas.

To remove panels from a canvas:

- Click  to remove a row of panels.

  OR

- Click  to remove a column of panels.

Merge panels

You can merge two or more panels to further enhance the design of your page.

Refrain from merging panels until you have added or removed all necessary panels. Adding and removing panels resets all changes you have made to the canvas.

Note: After a panel has been merged, that panel cannot be merged with another panel.

To merge panels:

1. Select the panels that you would like to merge.
2. Click  to merge the selected panels.

Set TAB order

A canvas's TAB order determines how OEP users navigate from one area of the page to another using their TAB key. A TAB order must be set for every canvas.

Tip: Complete all canvas design before setting TAB order. After you enter panel tab order mode, you cannot change the design of the canvas. Your only option is to save the canvas.

The default TAB order of a new canvas is from left to right for the first row, then left to right for the second row, and so on for each row defined in your canvas. TAB order begins with the number 0.

To set TAB order:

1. Click to switch to Panel Tab Order Mode.
2. Select each panel in the order in which you want to set the TAB order. You cannot clear your selections, so set TAB order with care. TAB order cannot be modified without recreating the canvas.
3. Apply the canvas.

Apply the canvas

You must set the TAB order for a canvas before applying the canvas to the page.

To apply the canvas:

- Click  to apply your canvas to the page.

Working with Panels

A panel represents an area within a canvas, much like a cell represents an area within a table. You can place any number of sections and UI controls into a panel. You can also convert a panel into a canvas, which in turn can contain additional panels and sections. Each UCW-enabled page or tab contains at least one panel, whether that panel exists on its own (as a single-panel canvas) or as part of a larger canvas.

In UCW design mode, panels are framed in green. The top-left corner of each panel is marked by a green triangle (◤) that, when right-clicked, displays a list of actions that you can perform on that panel. It also serves as a drop-spot for UI controls, captions, or sections that are dragged to it.

What would you like to do?

- Add a section to a panel
- Modify a panel's properties
- Convert a panel into to a canvas

Procedures

Add a section to a panel

You can add one or more sections to a panel.

To add a section to a panel:

- Right-click the panel's green triangle (◤) and select **Add Section**.

Modify a panel's properties

You can modify two properties of a panel: Admin ID and Label Width.

To modify a panel's properties:

1. Right-click the panel's green triangle (◤) and select **Edit Properties**.
2. In the Properties pane, edit the properties as required:
   - **Admin ID**. The Administration ID enables you to provide a user-friendly ID for the panel. Setting a value for this property is required. It accepts alphanumeric and underscore characters.
   - **Label Width**. This property determines the amount of space reserved for a UI control's label. If a control's label exceeds the specified width, the text wraps. If no width is specified, the label width for each control within the panel is set to the page's default label width. This property accepts numeric characters only.
3. Click  to apply your changes.
4. Click  on the UCW toolbar to save all configurations and exit design mode.

Working with Sections

Sections enable you to manage UI controls and captions as a group. They can be placed only within panels. You can place any number of UI controls into a section.

Placing controls and captions into a section is useful if you plan to dynamically perform actions on a group of controls/captions, rather than on each control/caption separately. For example, it might be useful to place certain company details, such as Tax ID and DUNS, in a section so you can more easily hide that information from users who don't need to see those details.

Sections also make it easy to move a group of UI controls and captions from one area of the page to another.

In design mode, sections are framed in blue. The top-left corner of each section is marked by a blue triangle ( ) that, when right-clicked, displays a list of actions that you can perform on that section. It also serves as a drop-spot for UI controls or captions that are dragged to it, and as a drag handle that enables you to drag the section to another location within the panel or to another panel.

What would you like to do?

- Add a section to a panel
- Hide a section
- Modify a section's properties
- Move a section
- Delete a section

Tip: Before configuring a section, click  to show all hidden UI elements. A section may have previously been hidden using Dynamic Forms Designer.

Procedures

Add a section to a panel

You can add one or more sections to a panel.

To add a section to a panel:

- Right-click the section's green triangle ( ) and select **Add Section**.

Modify a section's properties

You can modify one property of a section: Admin ID. The Administration ID enables you to provide a user-friendly ID for the section. It accepts alphanumeric and underscore characters.

To modify a section's properties:

1. Right-click the section's blue triangle ( ) and select Edit **Properties**.
2. In the **Properties** pane, edit the Admin ID as desired.

3. Click  to apply your changes.

4. Click  on the UCW toolbar to save all configurations and exit design mode.

Move a section

You can move a section by dragging it from one area of a page to another (though sections must always be placed within a panel). You can also move sections from one UCW-enabled tab to another.

Appropriate drop locations for the section include:

- A panel's drop-spot ( ).
- Another section's drop-spot ( ).

To move a section within a page:

- Select the section's blue triangle ( ) and drag it to the desired drop location.

To move a section to another UCW-enabled tab:

1. Select the section's blue triangle ( ).
2. Drag the section over the tab onto which you want it placed.
3. When the tab opens, drop the section onto the desired drop location.

Delete a section

Deleting a section removes all UI controls from that section. Because data binding for those UI controls may be removed as well, you may want to temporarily relocate the controls to another section (or panel) to save them for future use.

To delete a section:

- Right-click the section's blue triangle ( ) and select **Delete**.

Configuring UI Controls

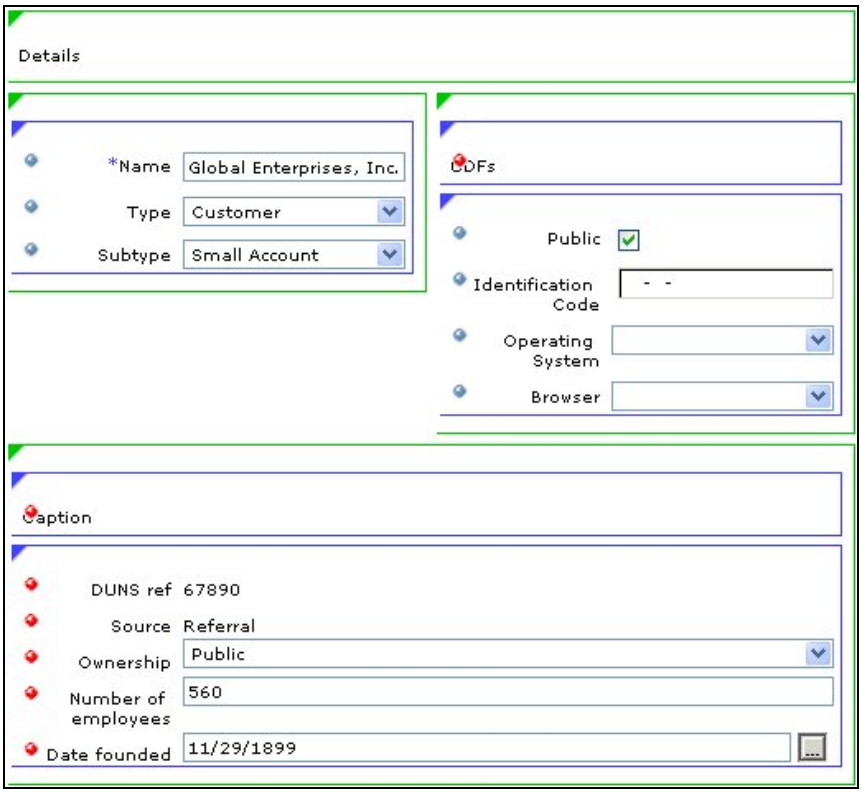
Using UCW, you can configure custom (toolbox) and default (intrinsic) UI controls on UCW-enabled pages. For example, you can change the text displayed for a caption, move an intrinsic text box control to another location, or add a custom drop-down list that is bound to an LBO property.

Toolbox controls include both user-interactive controls, such as drop-down lists and text- and date-formatted text boxes, and display-only controls, including captions and controls that list the selected lookup value or the current data. Toolbox controls do not exist until you add them to the page. Intrinsic controls include the controls, captions, and UDFs that appear on the page by default when you first install OEP.

Note: The company PowerPage and the individual PowerPage include some toolbox controls by default.

Whether toolbox or intrinsic, captions are UI text items that typically function as headings or names for panels, sections, groups of controls, toolbars, and records.

The following graphic illustrates intrinsic controls, default captions, and toolbox controls (including captions) in a custom layout on the company edit page during design mode. Intrinsic controls are identified by blue handles (). Toolbox controls, including captions, are identified by red handles (). (Default captions, such as the "Details" caption in this graphic, are not identified by handles.) To learn about a specific item in this example graphic, point to the item; click to see related configuration information.



You can also add functionality to controls by configuring action statements using Dynamic Forms Designer.

Note: Toolbars and HTML containers are not discussed in this book (other than their property descriptions). For information on the Toolbar toolbox control, see Configuring toolbars. For information on the HTML Container toolbox control, see Adding an HTML container control. .

What would you like to do?

- Learn about Intrinsic Controls (default controls)
- Learn about Toolbox Controls
- Add a control
- Show the controls panes
- Move a control to another location on the page
- Modify a control's properties
- Hide a section or control
- Modify a default caption
- Remove a control from the page
- Learn about control types and properties
- View a sample UI configuration procedure for UI controls
- View a sample UI configuration procedure for setting up UDFs

About Intrinsic Controls

The Intrinsic Controls pane appears when you access UCW for a selected page. When you first install OEP, the Intrinsic Controls pane is empty. Intrinsic Controls is populated with intrinsic (default) controls that you remove from the page, including UDFs that you activate (using OES) and captions.

During design mode, intrinsic controls are identified by blue control handles ( ).

Note: The company PowerPage and the individual PowerPage include some toolbox controls by default.

The following graphic illustrates the Intrinsic Controls pane. In this example, the intrinsic controls "Company Email" and several UDFs have been removed from the page.

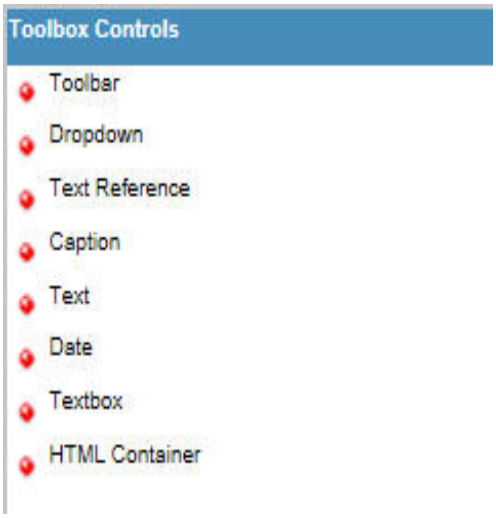
Modification of intrinsic controls' properties is limited to changing the related UI text, tooltip (text briefly displayed when users point to the control), and Admin ID (user-friendly identifier used in Dynamic Forms Designer). All data-binding information and referenced fields for intrinsic controls, including UDFs, are configured in OES. For basic information on UDFs, data-binding, and reference fields, see the OEAS Technical Reference.

Note: UDFs (User Defined Fields) appear in the Intrinsic Controls pane when they have been activated using OES Reference Table Administration. For information on creating and configuring UDFs, see Setting up UDFs.

About Toolbox Controls

The Toolbox Controls pane appears when you access UCW for a selected page. This pane lists toolbox control types available for adding custom controls to the page.

The following graphic illustrates the Toolbox Controls pane.



The following table describes each available toolbox control type.

Type (toolbox control)	Description	Data-bound?	User-editable?	Typical setting for the Mode property
Caption	UI text.	No	(N/A)	(N/A)
Date	Date-formatted text box for a user-entered date; contains a button that opens an interactive calendar as an alternative method for selecting a date. This control does not perform time zone conversions on dates.	Yes	Yes	Read/Write
Dropdown	Drop-down list of domain data values for	Yes	Yes	Read/Write

Type (toolbox control)	Description	Data-bound?	User-editable?	Typical setting for the Mode property
	user selection.			
HTML container	Placeholder UI container; functionality is configured with code written using Advanced View in Dynamic Forms Designer. To view a UI configuration example, see Adding an HTML container control.	Yes	Varies	Read/Write
Text	Value currently saved to the database for the data-bound property.	Yes	No; display-only control	Read Only
Text Reference	Value currently saved to the database for the data-bound property (currently saved domain data value from a drop-down list).	Yes	No; display-only control	Read Only
Textbox	Text box for user-entered data.	Yes	Yes	Read/Write
Toolbar	Toolbar. For information on adding toolbars, see Configuring toolbars.	No	(N/A)	(N/A)

During design mode, toolbox controls are identified by red control handles ( ).

There are no toolbox controls on the page when you first install OEP. Toolbox controls are created by dragging the desired toolbox control type from the Toolbox Controls pane to the page.

Note: The company PowerPage and the individual PowerPage include some toolbox controls by default.

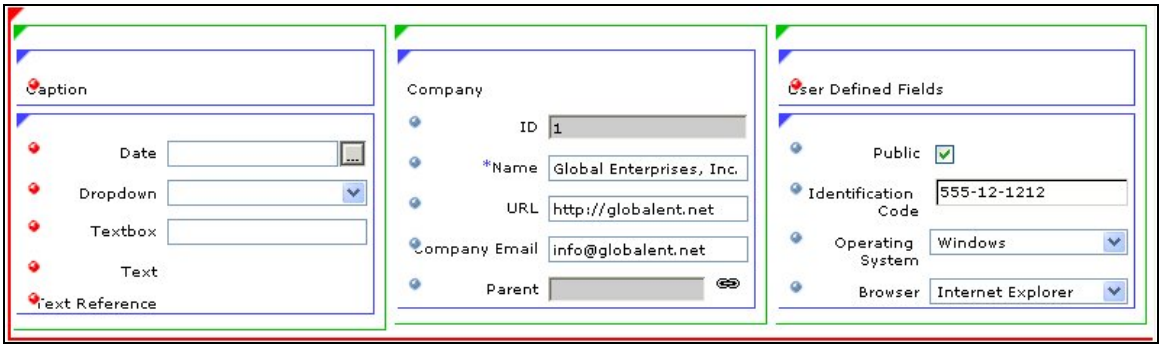
Modification of toolbox controls (other than captions) includes data-binding information. Modification of the Dropdown and Text Reference toolbox controls also includes referenced fields (for lookup values).

Adding Controls

You can add toolbox controls and intrinsic controls to UCW-enabled pages.

Toolbox controls enable you to add custom captions, drop-down lists, and text boxes (text- or date-formatted). You can also add display-only controls that list the current value (the selected lookup value or the user-typed entry). Outside design mode, there is no way to distinguish toolbox controls from intrinsic controls.

The following graphic illustrates toolbox controls and intrinsic controls during design mode. As shown in the graphic, toolbox controls are identified by red control handles (●) and intrinsic controls are identified by blue control handles (●). Each toolbox control (in the left-most panel) is labeled with its type; for example, the "Date" label indicates a Date toolbox control. To view a procedure for adding a particular type of toolbox control (such as the Dropdown toolbox control), click the toolbox control in the graphic.



Note: Toolbars and HTML containers are not discussed in this topic. For information on the Toolbar toolbox control, see Configuring toolbars. For information on the HTML Container toolbox control, see Adding an HTML container control.

Before you add UI controls, you might want to show dynamically hidden UI elements to ensure that the controls weren't already created.

What would you like to do?

- Learn about data-binding
- Add a custom caption
- Add a custom drop-down list
- Add a custom text box
- Add a custom date-formatted text box
- Add a custom display-only control
- Add an intrinsic control to the page

About data-binding

Note: Data-binding is not available for toolbox controls added to custom tabs on the Products page (product summary page) because this page is based on the retrieveList method. The methods retrieveList and retrieveCollection are not supported for data-binding in UCW.

When you add toolbox controls that have data-binding information, all page-related LBO properties (and fields, when applicable) are listed for your selection. To ensure that you have the correct property and field in mind, you may want to view their configuration in the Onyx Enterprise Dictionary (OED). For basic information on properties and fields, see OEAS Technical Reference and OEAS Technical Reference.

When you persist multiple controls that are bound to a single property, only the value entered for the most recent toolbox control is written to the record. Toolbox controls override any intrinsic controls, and more recently added toolbox controls override earlier toolbox controls. For example, if you persist two toolbox controls to the companyName property and retain the existing intrinsic control (persisted by default), only the value for the most recently added toolbox control is persisted to the database.

For an example of custom properties and fields for data-bound toolbox controls, see Adding UI controls.

Adding custom captions

You can add custom captions to the page. Captions are useful as titles for sections, panels, and groups of controls. Outside design mode, custom captions have the same appearance as default captions.

To add a custom caption to the page:

1. Drag **Caption** from the Toolbox Controls pane to the desired panel corner ( ), section corner ( ), or location near the handle of an existing control.

   The properties of the new control are automatically displayed in the Properties pane.

2. Configure the custom caption's properties.

   Properties for toolbox captions

Property	Description
ID	Identifier used for configuration in Advanced View for Dynamic Forms Designer. Not editable.
Admin ID	User-friendly identifier used for configuration of page behavior (Dynamic Forms Designer). Use alphanumeric and underscore characters only. The default Admin ID for a toolbox control begins with "toolbox_" and ends with a digit indicating order of toolbox control creation on the current tab. **Note:** Onyx recommends that you define unique Admin IDs to facilitate

Property	Description
	configuration. You can change the Admin ID property at any time.
Tooltip Text	Text briefly displayed when the user points to the selected control.
Label	UI text that is displayed on the page.

3. Click in the Properties pane to apply changes.
4. Click on the UCW toolbar to save all configurations and exit design mode.

Adding custom drop-down lists

You can add custom drop-down lists that are data-bound to LBO properties and that display lookup values for a selected field. Drop-down lists are useful when you want to limit user input to pre-configured options.

To add a custom drop-down list to the page:

1. Drag **Dropdown** from the Toolbox Controls pane to the desired panel corner ( ), section corner ( ), or location near the handle of an existing control.

   The properties of the new control are automatically displayed in the Properties pane.

2. Configure the custom control's properties.

   Properties for Dropdown toolbox controls

Property	Description
ID	Identifier used for configuration in Advanced View for Dynamic Forms Designer. Not editable.
Value	The current data value of the selected control. For example, if you select the "First Name" control for an individual record of John Smith, "John" is the entry. Not editable. Not available for captions.
Admin ID	User-friendly identifier used for configuration of page behavior (Dynamic Forms Designer). Use alphanumeric and underscore characters only. The default Admin ID for a toolbox control begins with "toolbox_" and ends with a digit indicating order of toolbox control creation on the current tab. **Note:** Onyx recommends that you define unique Admin IDs to facilitate configuration. You can change the Admin ID property at any time.
Tooltip Text	Text briefly displayed when the user points to the selected control.

Property	Description
Label	UI text that is displayed on the page.
Reference Data	(Heading only) - this group of properties applies to the Dropdown and Text Reference toolbox controls only.
-- Field Name	Field for which to display the value. For example, "incident.priority" indicates the "Priority" field for the incident object. Field names are defined in OES Reference Table Administration.
-- Parent ID	Parent field of the selected field name. For example, "3" indicates the sales category for the selected field (related to an incident object).
Data Binding	(Heading only) - this group of properties applies to data-bound toolbox controls only (Date, Dropdown, Text, Textbox, and Text Reference).
-- Object	Logical business object (LBO); page-specific OEAS data binding object. Not editable.
-- Property	LBO property (OEAS property) to which the selected control is bound. Properties are defined in OES Object Designer.
-- Mode	Sets the persistence mode for changed values. The selected option for this property determines whether data is persisted to the database when the selected control's value is changed—either by user input or by a "setValue" action configured in Dynamic Forms Designer. • **Read/Write** persists the changed value when the page is saved. (This option is selected by default for all data-bound controls.) • **Read Only** does not persist the changed value. The original value is retained. The Read Only setting is typical for the display-only controls Text and Text Reference. When you persist multiple controls that are bound to a single property, only the value entered for the most recent toolbox control is written to the record. Toolbox controls override any intrinsic controls, and more recently added toolbox controls override earlier toolbox controls. For example, if you persist two toolbox controls to the companyName property and retain the existing intrinsic control (persisted by default), only the value for the most recently added toolbox control is persisted to the database.

3. Click  in the Properties pane to apply changes.
4. Click  on the UCW toolbar to save all configurations and exit design mode.

Adding custom text boxes

You can add custom text boxes that are data-bound to LBO properties. Text boxes allow users to save free-text entries to the database.

For information about date-formatted text boxes, see Adding date-formatted text boxes.

To add a custom text box to the page:

1. Drag **Textbox** from the Toolbox Controls pane to the desired panel corner ( ), section corner ( ), or location near the handle of an existing control.

   The properties of the new control are automatically displayed in the Properties pane.

2. Configure the custom control's properties.

   Properties for Textbox toolbox controls

Property	Description
ID	Identifier used for configuration in Advanced View for Dynamic Forms Designer. Not editable.
Value	The current data value of the selected control. For example, if you select the "First Name" control for an individual record of John Smith, "John" is the entry. Not editable. Not available for captions.
Admin ID	User-friendly identifier used for configuration of page behavior (Dynamic Forms Designer). Use alphanumeric and underscore characters only. The default Admin ID for a toolbox control begins with "toolbox_" and ends with a digit indicating order of toolbox control creation on the current tab. **Note:** Onyx recommends that you define unique Admin IDs to facilitate configuration. You can change the Admin ID property at any time.
Tooltip Text	Text briefly displayed when the user points to the selected control.
Label	UI text that is displayed on the page.
Data Binding	(Heading only) - this group of properties applies to data-bound toolbox controls only (Date, Dropdown, Text, Textbox, and Text Reference).
-- Object	Logical business object (LBO); page-specific OEAS data binding object. Not editable.
-- Property	LBO property (OEAS property) to which the selected control is bound. Properties are defined in OES Object Designer.
-- Mode	Sets the persistence mode for changed values. The selected option for this property determines whether data is persisted to the database when the selected control's value is changed—either by user input or by a "setValue" action configured in Dynamic Forms Designer. • **Read/Write** persists the changed value when the page is saved. (This option is selected by default for all data-bound controls.)

Property	Description
	• **Read Only** does not persist the changed value. The original value is retained. The Read Only setting is typical for the display-only controls Text and Text Reference. When you persist multiple controls that are bound to a single property, only the value entered for the most recent toolbox control is written to the record. Toolbox controls override any intrinsic controls, and more recently added toolbox controls override earlier toolbox controls. For example, if you persist two toolbox controls to the companyName property and retain the existing intrinsic control (persisted by default), only the value for the most recently added toolbox control is persisted to the database.

3. Click  in the Properties pane to apply changes.
4. Configure validation of the text box's maximum length.
5. Click  on the UCW toolbar to save all configurations and exit design mode.

Adding custom date-formatted text boxes

You can add custom date-formatted text boxes that are data-bound to LBO properties. Date-formatted text boxes ensure that dates are correctly entered.

Note: See also Adding a date/time control (requires additional configuration).

To add a custom date-formatted text box to the page:

1. Drag **Date** from the Toolbox Controls pane to the desired panel corner ( ), section corner ( ), or location near the handle of an existing control.

   The properties of the new control are automatically displayed in the Properties pane.

2. Configure the custom control's properties.

   Properties for Date toolbox controls

Property	Description
ID	Identifier used for configuration in Advanced View for Dynamic Forms Designer. Not editable.
Value	The current data value of the selected control. For example, if you select the "First Name" control for an individual record of John Smith, "John" is the entry. Not editable. Not available for captions.
Admin ID	User-friendly identifier used for configuration of page behavior (Dynamic Forms Designer). Use alphanumeric and underscore characters only. The default Admin ID for a toolbox control begins with "toolbox_" and ends with a digit indicating order of toolbox control creation on the current tab. **Note:** Onyx recommends that you define unique Admin IDs to facilitate

Property	Description
	 configuration. You can change the Admin ID property at any time.
Tooltip Text	Text briefly displayed when the user points to the selected control.
Label	UI text that is displayed on the page.
Data Binding	(Heading only) - this group of properties applies to data-bound toolbox controls only (Date, Dropdown, Text, Textbox, and Text Reference).
-- Object	Logical business object (LBO); page-specific OEAS data binding object. Not editable.
-- Property	LBO property (OEAS property) to which the selected control is bound. Properties are defined in OES Object Designer.
-- Mode	Sets the persistence mode for changed values. The selected option for this property determines whether data is persisted to the database when the selected control's value is changed—either by user input or by a "setValue" action configured in Dynamic Forms Designer. • **Read/Write** persists the changed value when the page is saved. (This option is selected by default for all data-bound controls.) • **Read Only** does not persist the changed value. The original value is retained. The Read Only setting is typical for the display-only controls Text and Text Reference. When you persist multiple controls that are bound to a single property, only the value entered for the most recent toolbox control is written to the record. Toolbox controls override any intrinsic controls, and more recently added toolbox controls override earlier toolbox controls. For example, if you persist two toolbox controls to the companyName property and retain the existing intrinsic control (persisted by default), only the value for the most recently added toolbox control is persisted to the database.

3. Click  in the Properties pane to apply changes.
4. Click  on the UCW toolbar to save all configurations and exit design mode.

Adding custom display-only controls

You can add custom display-only controls to list the selected lookup value for a drop-down list or to list entered data for other types of controls.

To add a custom display-only control that lists the selected lookup value:

1. Drag **Text Reference** from the Toolbox Controls pane to the desired panel corner (), section corner (), or location near the handle of an existing control.

   The properties of the new control are automatically displayed in the Properties pane.
2. Configure the custom control's properties.

Properties for Text Reference toolbox controls

Property	Description
ID	Identifier used for configuration in Advanced View for Dynamic Forms Designer. Not editable.
Value	The current data value of the selected control. For example, if you select the "First Name" control for an individual record of John Smith, "John" is the entry. Not editable. Not available for captions.
Admin ID	User-friendly identifier used for configuration of page behavior (Dynamic Forms Designer). Use alphanumeric and underscore characters only. The default Admin ID for a toolbox control begins with "toolbox_" and ends with a digit indicating order of toolbox control creation on the current tab. **Note:** Onyx recommends that you define unique Admin IDs to facilitate configuration. You can change the Admin ID property at any time.
Tooltip Text	Text briefly displayed when the user points to the selected control.
Label	UI text that is displayed on the page.
Reference Data	(Heading only) - this group of properties applies to the Dropdown and Text Reference toolbox controls only.
-- Field Name	Field for which to display the value. For example, "incident.priority" indicates the "Priority" field for the incident object. Field names are defined in OES Reference Table Administration.
-- Parent ID	Parent field of the selected field name. For example, "3" indicates the sales category for the selected field (related to an incident object).
Data Binding	(Heading only) - this group of properties applies to data-bound toolbox controls only (Date, Dropdown, Text, Textbox, and Text Reference).
-- Object	Logical business object (LBO); page-specific OEAS data binding object. Not editable.
-- Property	LBO property (OEAS property) to which the selected control is bound. Properties are defined in OES Object Designer.
-- Mode	Sets the persistence mode for changed values. The selected option for this property determines whether data is persisted to the database when the selected control's value is changed—either by user input or by a "setValue" action configured in Dynamic Forms Designer. • **Read/Write** persists the changed value when the page is saved. (This option is selected by default for all data-bound controls.) • **Read Only** does not persist the changed value. The original value is retained. The Read Only setting is typical for the display-only controls Text and Text Reference. When you persist multiple controls that are bound to a single property, only the value

Property	Description
	entered for the most recent toolbox control is written to the record. Toolbox controls override any intrinsic controls, and more recently added toolbox controls override earlier toolbox controls. For example, if you persist two toolbox controls to the companyName property and retain the existing intrinsic control (persisted by default), only the value for the most recently added toolbox control is persisted to the database.

3. Click  in the Properties pane to apply changes.
4. Click  on the UCW toolbar to save all configurations and exit design mode.

To add a custom display-only control that lists the current data (not a lookup value):

1. Drag **Text** from the Toolbox Controls pane to the desired panel corner (), section corner (), or location near the handle of an existing control.

   The properties of the new control are automatically displayed in the Properties pane.

2. Configure the custom control's properties.

   Properties for Text toolbox controls.

Property	Description
ID	Identifier used for configuration in Advanced View for Dynamic Forms Designer. Not editable.
Value	The current data value of the selected control. For example, if you select the "First Name" control for an individual record of John Smith, "John" is the entry. Not editable. Not available for captions.
Admin ID	User-friendly identifier used for configuration of page behavior (Dynamic Forms Designer). Use alphanumeric and underscore characters only. The default Admin ID for a toolbox control begins with "toolbox_" and ends with a digit indicating order of toolbox control creation on the current tab. **Note:** Onyx recommends that you define unique Admin IDs to facilitate configuration. You can change the Admin ID property at any time.
Tooltip Text	Text briefly displayed when the user points to the selected control.
Label	UI text that is displayed on the page.
Data Binding	(Heading only) - this group of properties applies to data-bound toolbox controls only (Date, Dropdown, Text, Textbox, and Text Reference).
-- Object	Logical business object (LBO); page-specific OEAS data binding object. Not editable.
--	LBO property (OEAS property) to which the selected control is bound. Properties are

Property	Description
Property	defined in OES Object Designer.
-- Mode	Sets the persistence mode for changed values. The selected option for this property determines whether data is persisted to the database when the selected control's value is changed—either by user input or by a "setValue" action configured in Dynamic Forms Designer. • **Read/Write** persists the changed value when the page is saved. (This option is selected by default for all data-bound controls.) • **Read Only** does not persist the changed value. The original value is retained. The Read Only setting is typical for the display-only controls Text and Text Reference. When you persist multiple controls that are bound to a single property, only the value entered for the most recent toolbox control is written to the record. Toolbox controls override any intrinsic controls, and more recently added toolbox controls override earlier toolbox controls. For example, if you persist two toolbox controls to the companyName property and retain the existing intrinsic control (persisted by default), only the value for the most recently added toolbox control is persisted to the database.

3. Click  in the Properties pane to apply changes.
4. Click  on the UCW toolbar to save all configurations and exit design mode.

Adding intrinsic controls

You can add intrinsic controls to the page when those controls were previously removed from the page (or, in the case of UDFs, were either newly activated or removed). All data-binding information and referenced fields for intrinsic controls are configured in OES. For more information, see *OEAS Technical Reference* and *OEAS Technical Reference*.

Note: UDFs (User Defined Fields) appear in the Intrinsic Controls pane during design mode when they have been activated using OES Reference Table Administration. For more information on creating and configuring UDFs, see Setting up UDFs.

Note: UI text for intrinsic controls can also be modified using the UI Text Editor.

To add an intrinsic control to the page:

1. Drag the removed control from the Intrinsic Controls pane to the desired panel corner (), section corner (), or location near the handle of an existing control.

   The properties of the moved control are automatically displayed in the Properties pane.
2. Configure the intrinsic control's properties.

Properties for intrinsic controls (including captions and UDFs).

Property	Description
ID	Identifier used for configuration in Advanced View for Dynamic Forms Designer. Not editable.
Value	The current data value of the selected control. For example, if you select the "First Name" control for an individual record of John Smith, "John" is the entry. Not editable.
Admin ID	User-friendly identifier used for configuration of page behavior (Dynamic Forms Designer). Use alphanumeric and underscore characters only. **Note:** Onyx recommends that you define unique Admin IDs to facilitate configuration. You can change the Admin ID property at any time.
Tooltip Text	Text briefly displayed when the user points to the selected control.
Label	UI text that is displayed on the page. Available for data-bound controls.
Caption	UI text that is displayed on the page. Available for default captions only.

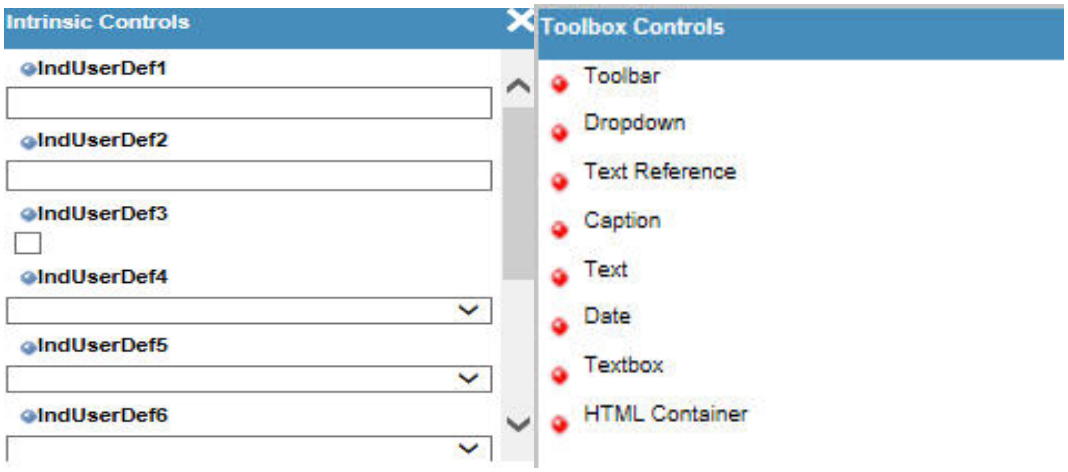
3. Click in the Properties pane to apply changes.
4. Click on the UCW toolbar to save all configurations and exit design mode.

Showing the Controls Panes

You can show and hide the controls panes. Controls panes include Intrinsic Controls, Toolbox Controls, and Properties. Use controls panes to configure UI controls on the page.

Controls panes are automatically docked when you enter design mode.

Example of the Intrinsic Controls, Toolbox Controls, and Properties panes.



To show or hide the Intrinsic Controls, Toolbox Controls, and Properties panes:

- Click the related button on the UCW toolbar, according to the following table.

To:	Click this button on the UCW toolbar:
Disable docking (hide all panes)	
Show or hide the Toolbox Controls pane when docking is disabled	
Show or hide the Intrinsic Controls pane when docking is disabled	
Enable docking (display all panes, docked)	

Moving Controls

You can move controls from one location on the page to another. Default captions are kept with moved sections or controls.

To move a control to another location on the current tab:

- Drag the control's handle (or) to the desired panel corner (), section corner (), or location near the handle of an existing control.

To move a control to another tab:

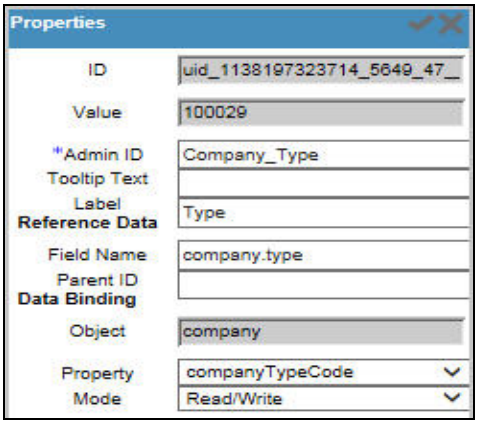
- While dragging the control's handle (or), point to the desired tab to display the tab; continue dragging the control's handle to the desired panel corner (), section corner (), or location near the handle of an existing control.

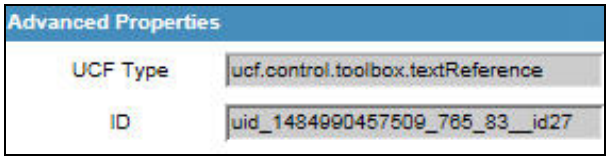
Modifying Controls

You can modify controls on the page by configuring their properties.

Note: UI text for intrinsic controls can also be modified using the UI Text Editor.

The following graphics illustrate the Properties pane and the Advanced Properties pane. In this example, the control is data-bound to the sourceId property on the company object and is configured to reference the company.source field. The UCF type "ucf.control.toolbox.textReference" indicates a Text Reference toolbox control.





Note: You can add functionality to controls by configuring action statements using Dynamic Forms Designer.

To modify a control:

1. If the desired control is displayed in the Intrinsic Controls pane, move the control to the page for configuration.
2. Display the control's properties in the Properties pane:
   - Right-click the control's handle (or) and then select Properties.

     OR

   - Click the control's handle (or) (if the Properties pane is already displayed).

     Note: For information on control properties, see Control properties reference.

Note: Properties differ by control type. To view the control's control type (for example, to determine whether a toolbox control is a Text toolbox control or a Text Reference toolbox control), right-click the control's handle (or) and then select Advanced Properties.

3. In the Properties pane, change the desired entries and then click to apply changes.
4. Click on the UCW toolbar to save all configurations and exit design mode.

Hiding Sections and Controls

You can hide sections and controls from users who log on to OEP with the selected profile.

Unlike deleted sections and controls, hidden sections and controls can be shown during design mode.

Note: You can also hide sections, controls, and other UI elements dynamically, by defining conditions for the hide action. For an example, see Changing pages dynamically.

What would you like to do?

- Hide a section
- Hide a control

Hiding sections

You can hide sections from users who log on to OEP with the selected profile. When you hide sections, all controls in the sections are also hidden.

Because this procedure uses the Load event, you must reload the page after completing the procedure to see the changes.

To hide a section:

1. Click for the current page to enter design mode.
2. Note the Admin ID of the section you want to hide. (To view the Admin ID of a section,right-click the section's corner ( ) and then select **Properties**. Example Admin IDs for sections: Section_ Company and Section_CustomControls.)

   Dynamic Forms Designer identifies UI elements by their Admin IDs. Onyx recommends that you define unique Admin IDs to facilitate configuration. You can change the **Admin ID** property at any time.

3. Click on the UCW toolbar to display Dynamic Forms Designer.
4. In the Events pane, expand **Page** and then select **Load**.

5. Click  to add an action statement.
6. In the **Description** text box at the top of the Action Designer pane, type a description. For example, "Hide the ABC section."
7. In the **Object (Action)** drop-down list located in the Actionsection, expand **Section**, expand the desired section (identified by Admin ID), and then select **Hide Section**.

   The **Object (Action)** text box now displays the selected section and the hide action (example: "Section_Company (Hide Section)").

8. Click  to confirm changes and then close Dynamic Forms Designer.
9. Click  on the UCW toolbar to save all configurations and exit design mode.

Hiding controls

You can hide controls from users who log on to OEP with the selected profile. When you hide sections, all controls in the sections are also hidden.

Because this procedure uses the Load event, you must reload the page after completing the procedure to see the changes.

To hide a control:

1. Click  for the current page to enter design mode.
2. Note the Admin ID of the control you want to hide. (To view the Admin ID of a control,right-click the control's handle (or)and then select **Properties**. Example Admin IDs for controls: Tax_ID and toolbox_Birthday.)

    **Note:** Dynamic Forms Designer identifies UI elements by their Admin IDs. Onyx recommends that you define unique Admin IDs to facilitate configuration. You can change the **Admin ID** property at any time.

3. Click  on the UCW toolbar to display Dynamic Forms Designer.
4. In the Events pane, expand **Page** and then select **Load**.
5. Click  to add an action statement.
6. In the **Description** text box at the top of the Action Designer pane, type a description. For example, "Hide the ABC control."
7. In the **Object (Action)** drop-down list located in the Actionsection, expand **Section**, expand the desired control (identified by Admin ID), and then select **Hide**.

The **Object (Action)** text box now displays the selected control and the hide action (example: "Tax_ID (Hide)").

8. Click to confirm changes and then close Dynamic Forms Designer.
9. Click on the UCW toolbar to save all configurations and exit design mode.

Modifying Default Captions

Captions that appear on UCW-enabled pages by default are not identified by control handles. These captions cannot be individually moved, deleted, or removed from the page; however, you can modify default captions by changing or removing the displayed text.

Example graphic illustrating default captions

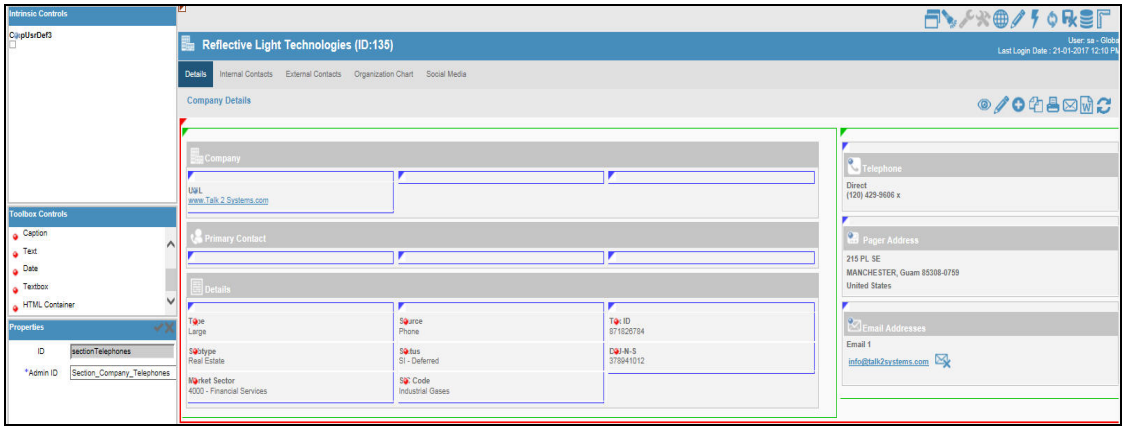
The following graphic illustrates default captions (and one custom caption) as displayed during design mode on the company PowerPage. Default captions are not identified by control handles. The blue control handles ( ) identify default controls while the red control handles ( ) identify custom controls, including the custom caption ("Custom Caption" on a gray background).

Note: On pages other than the PowerPage (company or individual), section titles and embedded captions are displayed on a white background.

Types of default captions in this graphic include:

- Record identifier ("Global Enterprises, Inc." on white background)
- Toolbar caption ("Company Details" on blue background)
- Section titles ("Company" and "Primary Contact" on gray background)
- Captions embedded in controls ("Telephone" and "Billing Address" on gray background)

Note: For more information on editing record identifiers and other UI text, see Configuring UI text.



Note: When you remove a control with an embedded default caption, the control and caption are both placed in the Intrinsic Controls pane. When you delete a section containing default caption(s), the default caption(s) are removed from memory along with the rest of the contents. You can recreate the caption using the Caption toolbox control type. For more information on moving or removing controls, see Configuring UI controls.

To remove the text displayed for a default caption:

1. Display the caption's properties in the Properties pane:
   - Right-click the default caption and then select Properties.

     OR
   - Click the default caption (if the Properties pane is already displayed).
2. Clear the Caption text box in the Properties pane.
3. Click  in the Properties pane to apply changes.

   The default caption is identified on the page by its Admin ID property. For example, if you remove the "Edit Company" string from the default caption displayed on the upper left of the company edit page, the default caption is identified by "Caption_Edit_Company" (its default Admin ID property). The Admin ID property is visible during design mode only; OEP users do not see the Admin ID property.
4. Click  on the UCW toolbar to save all configurations and exit design mode.

To modify a default caption:

1. Display the caption's properties in the Properties pane:
   - Right-click the default caption and then select Properties.

     OR
   - Click the default caption (if the Properties pane is already displayed).
2. Configure the caption's properties.

Property	Description
Admin ID	User-friendly identifier used for configuration of page behavior (Dynamic Forms Designer). Use alphanumeric and underscore characters only. Onyx recommends that you define unique Admin IDs to facilitate configuration. You can change the Admin ID property at any time.
Tooltip Text	Text briefly displayed when the user points to the selected control.

Property	Description
Caption	UI text that is displayed on the page.

3. Click in the Properties pane to apply changes.
4. Click on the UCW toolbar to save all configurations and exit design mode.

Removing Controls

You can remove selected controls from the page. You can also remove all controls from a section, or all controls from the page, one tab at a time.

Removed toolbox controls ( ) are deleted. Removed intrinsic controls ( ), including UDFs, are retained in the Intrinsic Controls pane, along with their embedded default captions, if any.

Note: When you delete a section containing default caption(s), the default caption(s) are removed from memory along with the rest of the contents. You can recreate the caption using the Caption toolbox control type. For more information on default captions, see Modifying default captions.

To remove a single control from the page:

- Right-click the control's handle (or) and then select **Delete** (for a toolbox control) or **Remove Intrinsic Control** (for an intrinsic control). If the control is an intrinsic control, you can also drag the control's handle ( ) to the Intrinsic Controls pane.

To remove all controls in a section (deletes the section):

- Right-click the section corner ( ) and then select **Delete**. For more information on sections, see Working with sections.

To remove all controls from the page (current tab only):

- Right-click the top-most canvas corner ( ) and then select **Revert to Panel** to change the canvas into an empty panel. For more information on reverting canvases to panels, see Working with canvases.

Validating Maximum Length (Textbox toolbox controls)

You can validate data entered in **Textbox** toolbox controls to ensure that the data does not exceed the maximum length.

This procedure involves configuration of a validation message that is displayed when the OEP user attempts to save the record or change focus (by selecting another control, for example) after entries greater than the maximum length.

To validate length of entered data for a Textbox toolbox control:

1. Note the **Admin ID** property for the toolbox control: In design mode, right-click the control's handle ( ) and then select **Properties**.

   Note: Dynamic Forms Designer identifies UI elements by their Admin IDs. Onyx recommends that you define unique Admin IDs to facilitate configuration. You can change the **Admin ID** property at any time.

2. Click on the UCW toolbar to display Dynamic Forms Designer.
3. In the Events pane, expand the **Page** object, and then select the **Validate** event.
4. Click to add an action statement.
5. In the **Description** text box at the top of the Action Designer pane, type a description of the statement. For example: "Validate length of Textbox toolbox control."
6. In the Conditions section of the Action Designer pane, configure the condition as follows.
   - From the **Object** drop-down list, expand **Control**, expand the noted Admin ID for the **Textbox** toolbox control (for example, expand "toolbox_num"), and then select **Get Value**.
   - From the **Operator** drop-down list, select **Length Greater Than**; in the Value text box, type the desired maximum number of characters.
   - Click in the Summary section to add the condition to the action statement.
7. In the Action section of the Action Designer pane, configure the action as follows.
   - From the **Object (Action)** drop-down list, expand **Common**, expand **UI**, and then select **Show Validation Message Box**.
   - Configure the sources and values for the validation message box using the following entries.

(Row)	For the Source, do this:	For the Value, do this:
Title	Select **User Defined**	Type: Error
Message	Select **User**	Type a message indicating the control's label. For example: The

(Row)	For the Source, do this:	For the Value, do this:
	Defined	CustomControl text box contains more than the maximum number of characters.
Icon Type	Select **User Defined**	Select **Exclamation**

- Select the **Stop on Execute**? check box to prevent subsequent statements within the Validate event from executing if the control's value exceeds the maximum length (that is, if the conditions are met).

For more information on the Stop on Execute? check box, see Actions and conditions.

8. Click to confirm changes.
9. Copy this statement to the control's Change event:
   a. Click to clone the statement.
   b. In the Events pane, expand the **Control** object and then expand the noted Admin ID for the **Textbox** toolbox control (for example, expand "toolbox_num").
   c. Drag the cloned statement's handle ( ) to the **Change** event ( ) displayed for the **Textbox** toolbox control.

   The **Change** event ( ) is displayed.
10. Close Dynamic Forms Designer and then click on the UCW toolbar to save all configurations and exit design mode.

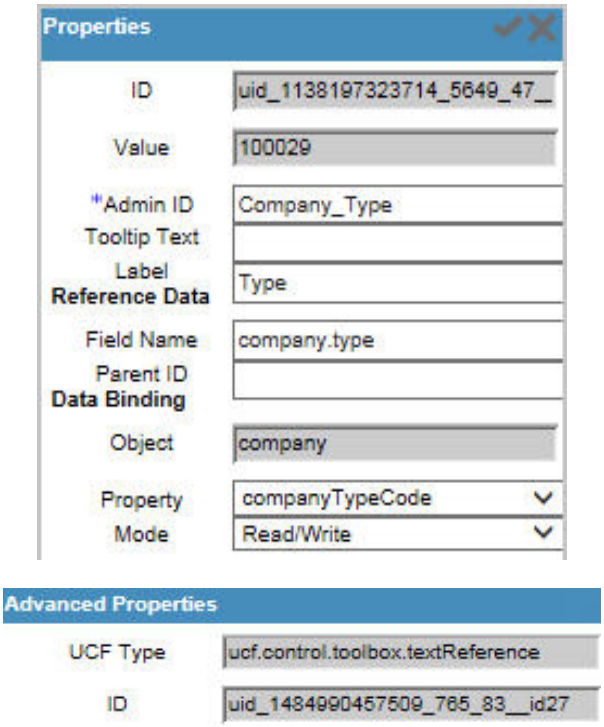
Control Properties Reference

This topic describes properties for toolbox controls and intrinsic controls. Control properties appear in the Properties pane while advanced control properties appear in the Advanced Properties pane.

Required properties are indicated with asterisks.

Example of Properties and Advanced Properties

The following graphics illustrate the Properties pane and the Advanced Properties pane. In this example, the control is data-bound to the sourceId property on the company object and is configured to reference the company.source field. The UCF type "ucf.control.toolbox.textReference" indicates a Text Reference toolbox control.



Properties for toolbox controls

Toolbox (custom) controls include toolbox captions.

Property	Description
ID	Identifier used for configuration in Advanced View for Dynamic Forms Designer. Not editable.
Value	The current data value of the selected control. For example, if you select the "First Name" control for an individual record of John Smith, "John" is the entry. Not editable. Not available for captions.
Admin ID	User-friendly identifier used for configuration of page behavior (Dynamic Forms Designer). Use alphanumeric and underscore characters only.

Property	Description
	The default Admin ID for a toolbox control begins with "toolbox_" and ends with a digit indicating order of toolbox control creation on the current tab.  **Note:** Onyx recommends that you define unique Admin IDs to facilitate configuration. You can change the Admin ID property at any time.
Association ID	Identification code of related toolbar configuration. Available for Toolbar toolbox controls only. This entry maps the selected Toolbar toolbox control on the page with the specified toolbar configuration created in Navigation Designer. For information on adding toolbars, see Configuring toolbars.
Tooltip Text	Text briefly displayed when the user points to the selected control.
Label	UI text that is displayed on the page.
Reference Data	(Heading only) - this group of properties applies to the Dropdown and Text Reference toolbox controls only.
-- Field Name	Field for which to display the value. For example, "incident.priority" indicates the "Priority" field for the incident object. Field names are defined in OES Reference Table Administration.
-- Parent ID	Parent field of the selected field name. For example, "3" indicates the sales category for the selected field (related to an incident object).
Data Binding	(Heading only) - this group of properties applies to data-bound toolbox controls only (Date, Dropdown, Text, Textbox, and Text Reference).
-- Object	Logical business object (LBO); page-specific OEAS data binding object. Not editable.
-- Property	LBO property (OEAS property) to which the selected control is bound. Properties are defined in OES Object Designer.
-- Mode	Sets the persistence mode for changed values. The selected option for this property determines whether data is persisted to the database when the selected control's value is changed—either by user input or by a "setValue" action configured in Dynamic Forms Designer. • **Read/Write** persists the changed value when the page is saved. (This option is selected by default for all data-bound controls.) • **Read Only** does not persist the changed value. The original value is retained. The Read Only setting is typical for the display-only controls Text and Text Reference. When you persist multiple controls that are bound to a single property, only the value entered for the most recent toolbox control is written to the record. Toolbox controls override any intrinsic controls, and more recently added toolbox controls override earlier toolbox controls. For example, if you persist two toolbox controls to the companyName property and retain the existing intrinsic control (persisted by default), only the value for the most recently added toolbox control is persisted to the database.

Properties for intrinsic controls

Intrinsic controls includes default captions and UDFs (User Defined Fields). UDFs appear in the Intrinsic Controls pane during design mode when they have been activated using OES Reference Table Administration. For more information about creating and configuring UDFs, see Setting up UDFs.

Note: UI text for intrinsic controls can also be modified using the UI Text Editor.

Property	Description
ID	Identifier used for configuration in Advanced View for Dynamic Forms Designer. Not editable.
Value	The current data value of the selected control. For example, if you select the "First Name" control for an individual record of John Smith, "John" is the entry. Not editable.
Admin ID	User-friendly identifier used for configuration of page behavior (Dynamic Forms Designer). Use alphanumeric and underscore characters only. **Note:** Onyx recommends that you define unique Admin IDs to facilitate configuration. You can change the Admin ID property at any time.
Tooltip Text	Text briefly displayed when the user points to the selected control.
Label	UI text that is displayed on the page. Available for data-bound controls.
Caption	UI text that is displayed on the page. Available for default captions only.

Advanced Properties

Advanced properties are not editable.

Property	Description
UCF Type	Type of control. Examples: "ucf.control.toolbox.text" or "ucf.control.simple.dropdown"
ID	Identifier used for configuration in Advanced View for Dynamic Forms Designer.

Configuring Page Navigation

Using Navigation Designer, you can modify how your users navigate through OEP to perform their tasks by configuring the Header Bar, Navigation Bar, toolbars, and tabs.

To launch Navigation Designer:

- Click  to enter design mode and then click  on the UCW toolbar to display Navigation Designer.

What would you like to do?

- Configure the Header Bar
- Configure the Navigation Bar
- Configure a toolbar
- Configure a tab

Configuring the Header Bar

The Header Bar contains buttons that enable OEP users to navigate to other OEP pages, such as the Customer PowerPage and the Search page. The Header Bar is configured as part of the main application frame.

The following graphic shows a sample Header Bar with a custom Header Bar button that, when clicked, opens a Google search page. To learn how to add a custom button like this one, see in the sample UI configuration Adding a header bar button.

What would you like to do?

- Create images for a Header Bar button
- Add a button to the Header Bar
- Modify a button on the Header Bar
- Move a button on the Header Bar
- Hide a button on the Header Bar
- Delete a button from the Header Bar

Procedures

Creating images for Header Bar buttons

You can create images for Header Bar buttons.

To create images for a Header Bar button:

1. With your favorite graphics software, use the following guidelines to create an image for each button state: Selected and Unselected.

Attribute	Guideline
Format	Graphics Interchange Format (GIF).
Size	31 pixels wide x 22 pixels tall.

   The Header Bar button properties enable you to specify the file name of the image representing each button state.

2. Copy the image files to the following directory on your OEP Web server: YourOEPwebsite/ucf/data/custom/yourCompany (Onyx recommends creating a company-specific subdirectory in the custom directory).

Adding buttons to the Header Bar

You can add buttons to the Header Bar. To view a sample configuration for a new Header Bar button, see Adding a header bar button.

To add a button to the Header Bar:

1. Create images for the Header Bar button and place the image files in the appropriate UCW directory.
2. Click  above the Header Bar to enter design mode for the main application frame and then click  on the UCW toolbar to display Navigation Designer.
3. From the Control drop-down list, select Header Bar to display the existing Header Bar buttons.
4. Specify the insertion point for the new button:
   - To place it last (right-most button on the Header Bar), click Header Bar in the Items pane.
   - To choose another position for the new button, expand Header Bar in the Items pane and then click the button before the desired insertion point.
5. Click  to add the button.
6. Configure the button's properties.

Properties for Header Bar buttons

Property	Description
ID	Unique identifier of the Header Bar button. Use alphanumeric characters and underscores only. The identifier cannot be changed once set. **Note:** For IDs entered during configuration of the Header Bar and the Navigation Bar, case-sensitivity is not assessed; Navigation Designer does not accept IDs that differ from an existing ID only in case. For example, if the ID "iconOn" already exists, a new ID "iconON" is not accepted.
Caption	Text identifying the button on the page and in Navigation Designer.
Tooltip	Text briefly displayed when the user points to the button.
URL	(Available for custom buttons only.) Uniform Resource Locator (URL) identifying the page to be retrieved.
UI Resource	Hides the selected button if the user does not have access to the specified UI resource.
Target	(Available for custom buttons only.) The target frame or window for the page retrieved by the URL. Available selections are **Open URL into a new window** and **Open URL into the Data Area frame**.
Target Arguments	(Available for custom buttons only.) A string of values to pass directly to the sFeatures parameter of the DHTML window.open method. Refer to MSDN's DHTML documentation for the supported options of this parameter.
Selected Image	Specifies the file name and path of the image representing the selected state of the Header Bar button. Image files for custom buttons must be stored in the displayed directory (YourOEPwebsite/ucf/data/custom). Onyx recommends creating a company-specific subdirectory in the custom directory. **Note:** Image files for default buttons are stored in the *YourOEPwebsite*/images/headerbar directory.
Unselected Image	Specifies the file name and path of the image representing the unselected state of the Header Bar button. Image files for custom buttons must be stored in the displayed directory (*YourOEPwebsite*/ucf/data/custom). Onyx recommends creating a company-specific subdirectory in the custom directory. **Note:** Image files for default buttons are stored in the *YourOEPwebsite*/images/headerbar directory.

7. Click  to confirm changes and close Navigation Designer.
8. Click  on the UCW toolbar to save all configurations and exit design mode.

Modifying buttons on the Header Bar

You can modify the properties of existing Header Bar buttons, such as their captions, tooltips, required UI resources, and image files. Custom buttons have additional properties available for modification.

To modify a button on the Header Bar:

1. Click  above the Header Bar to enter design mode for the main application frame and then click  on the UCW toolbar to display Navigation Designer.
2. From the **Control** drop-down list, select **Header Bar** to display the existing Header Bar buttons.
3. In the Items pane, click the desired button to display its properties in the Properties pane and then change the properties as desired.

   Properties for Header Bar buttons

Property	Description
ID	Unique identifier of the Header Bar button. Use alphanumeric characters and underscores only. The identifier cannot be changed once set.  **Note:** For IDs entered during configuration of the Header Bar and the Navigation Bar, case-sensitivity is not assessed; Navigation Designer does not accept IDs that differ from an existing ID only in case. For example, if the ID "iconOn" already exists, a new ID "iconON" is not accepted.
Caption	Text identifying the button on the page and in Navigation Designer.
Tooltip	Text briefly displayed when the user points to the button.
URL	(Available for custom buttons only.) Uniform Resource Locator (URL) identifying the page to be retrieved.
UI Resource	Hides the selected button if the user does not have access to the specified UI resource.
Target	(Available for custom buttons only.) The target frame or window for the page retrieved by the URL. Available selections are **Open URL into a new window** and **Open URL into the Data Area frame**.
Target Arguments	(Available for custom buttons only.) A string of values to pass directly to the sFeatures parameter of the DHTML window.open method. Refer to MSDN's DHTML documentation for the supported options of this parameter.
Selected	Specifies the file name and path of the image representing the selected state of the

Property	Description
Image	Header Bar button. Image files for custom buttons must be stored in the displayed directory (*YourOEPwebsite*/ucf/data/custom). Onyx recommends creating a company-specific subdirectory in the custom directory. **Note:** Image files for default buttons are stored in the *YourOEPwebsite*/images/headerbar directory.
Unselected Image	Specifies the file name and path of the image representing the unselected state of the Header Bar button. Image files for custom buttons must be stored in the displayed directory (*YourOEPwebsite*/ucf/data/custom). Onyx recommends creating a company-specific subdirectory in the custom directory. **Note:** Image files for default buttons are stored in the *YourOEPwebsite*/images/headerbar directory.

4. Click to confirm changes and close Navigation Designer.
5. Click on the UCW toolbar to save all configurations and exit design mode.

Moving buttons on the Header Bar

You can change the position of buttons on the Header Bar. For example, you can move the third button from the left to be the first button from the left.

To move a button on the Header Bar:

1. Click above the Header Bar to enter design mode for the main application frame and then click on the UCW toolbar to display Navigation Designer.
2. From the Control drop-down list, select Header Bar to display the existing Header Bar buttons.
3. In the Items pane, drag the button's handle (▪) to the desired position on the Header Bar.

   The highest position in Navigation Designer translates to the left-most position on the Header Bar.
4. Click to confirm changes and close Navigation Designer.
5. Click on the UCW toolbar to save all configurations and exit design mode.

Hiding buttons on the Header Bar

You can hide buttons on the Header Bar from users who log on to OEP with the selected profile. You can also hide buttons based on certain conditions. For more information on conditions, see Configuring page behavior.

Because this procedure uses the Load event, you must reload the page after completing the procedure to see the changes.

To hide a button on the Header Bar:

1. Click  above the Header Bar to enter design mode for the main application frame and then click  on the UCW toolbar to display Dynamic Forms Designer.
2. In the Events pane, expand **Page** and then select **Load**.
3. Click  to add an action statement.
4. In the **Description** text box at the top of the Action Designer pane, type a description. For example, "Hide the ABC Header Bar button."
5. Configure the action as follows:
   a. From the **Object (Action)** drop-down list, expand **Control**, expand **Header_Bar**, and then select **Hide Item**.
   b. From the **Source** drop-down list, select **User Defined**; from the **Value** drop-down list, select the desired button. (Note: Item ID values for this item use the Caption property from Navigation Designer.)
6. Click  to confirm changes and then close Dynamic Forms Designer.
7. Click  on the UCW toolbar to save all configurations and exit design mode.

Deleting buttons from the Header Bar

You can delete custom buttons from the Header Bar. Default Header Bar buttons cannot be deleted, but you can hide buttons from users who log on to OEP with the selected profile.

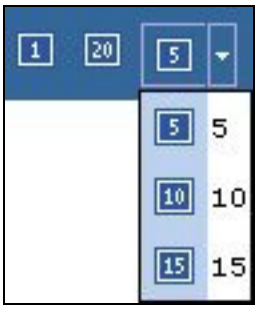
To delete a button from the Header Bar:

1. Click  above the Header Bar to enter design mode for the main application frame and then click  on the UCW toolbar to display Navigation Designer.
2. From the **Control** drop-down list, select **Header Bar** to display the existing Header Bar buttons.
3. In the Items pane, click the desired button.
4. Click  to delete the selected button.
5. Click  to confirm changes and close Navigation Designer.
6. Click  on the UCW toolbar to save all configurations and exit design mode.

Configuring Toolbars

Toolbars can contain buttons and other toolbar items that enable OEP users to perform specific actions on a page, such as saving entered data, printing the page, or cloning (duplicating) the record. You can configure toolbars on UCW-enabled pages.

The following graphic shows a custom toolbar with the following types of custom toolbar items: Button, MultiState, and Action Menu. The Action Menu toolbar item contains three values. To learn how to add toolbar items like those depicted here, see the sample UI configuration Adding toolbar buttons.



What would you like to do?

- Add a toolbar
- Changing toolbar names
- Create images for a toolbar item value
- Add a button or other toolbar item to a toolbar
- Add functionality to a toolbar item
- Modify a toolbar item
- Move a toolbar item on a toolbar
- Hide a toolbar item on a toolbar
- Delete a toolbar item from a toolbar
- Hide a toolbar
- Delete a toolbar

Procedures

Adding toolbars

You can add toolbars to UCW-enabled pages. To view a sample configuration for a new toolbar, see Adding a toolbar.

Note: If you suspect that a custom toolbar was added previously even though you don't see it on the page, click on the UCW toolbar to show all hidden UI elements. Sections containing toolbars can be hidden dynamically (through Dynamic Forms Designer) or otherwise.

To add a toolbar:

1. Click for the current page to enter design mode and then click on the UCW toolbar to display Navigation Designer.
2. From the **Control** drop-down list, select **Toolbar Controls** to display the existing toolbars for the current page.
3. In the Items pane, click **Toolbar Controls**.
4. Click to add a toolbar.
5. In the **ID** text box, type a unique identifier for the toolbar, using alphanumeric characters and underscores only. Note the identifier.
6. (Optional) Add buttons and other toolbar items to the toolbar.
7. Click to confirm changes and close Navigation Designer.
8. Drag **Toolbar** from the Toolbox Controls pane to the desired section or panel of the page. (If docking is disabled, click on the UCW toolbar to display Toolbox Controls.)
9. Using the Properties pane, configure the toolbar's properties.
   a. In the **Association ID** drop-down list, select the ID that you defined in Navigation Designer.
   b. In the **Title** text box, type the desired title.
   c. Click to apply your changes.

   The new toolbar is displayed on the page; the configured title is displayed on the left of the toolbar.
10. Click on the UCW toolbar to save all configurations and exit design mode.

Changing toolbar names

You can change toolbar names for custom and default toolbars on UCW-enabled pages. Names for custom toolbars are specified by the Title property of the toolbar while names for default toolbars are specified by the default caption's Caption property.

Each toolbar name appears on the left of the related toolbar.

Note: You can also change a default toolbar's name (caption) using UI Text Editor.

To change the name of a custom toolbar:

1. Click for the current page to enter design mode.
2. Right-click the toolbar toolbox control's handle ( ) and select **Properties**.
3. In the Properties pane, type the desired name in the **Title** text box and then click to apply changes.
4. Click on the UCW toolbar to save all configurations and exit design mode.

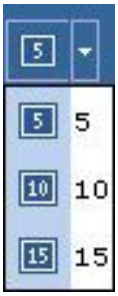
To change the name of a default toolbar:

1. Click for the current page to enter design mode.
2. Right-click the existing toolbar name (default caption) and select **Properties**. (In design mode, default toolbars with no defined names are identified by their Admin IDs.)
3. In the Properties pane, type the desired name in the **Caption** text box and then click to apply changes.
4. Click on the UCW toolbar to save all configurations and exit design mode.

Creating images for toolbar item values

You can create images for each value in a toolbar item.

The following graphic illustrates an expanded drop-down list (**Action Menu** toolbar item) with the captions "5", "10", and "15". The displayed images correspond to the "Normal" state.



To create images for a toolbar item value:

1. With your favorite graphics software, use the following guidelines to create an image for each state of the toolbar item value.

Attribute	Guideline
Format	Graphics Interchange Format (GIF).
Size	23 pixels wide x 22 pixels tall.
States	• Normal state, represented by the digit 0 • Mouseover state, represented by the digit 2 • Clicked state, represented by the digit 3 • Disabled state, represented by the digit 4 To view example images for each state, see Button standards.
File name	Use standard beginnings and endings: Begin each file name with the phrase "icon" and end each file name with the appropriate state digit. Example for the Clicked state of a custom image: "iconCustom3.gif"

2. The properties in Navigation Designer enable you to specify the file name of the image representing the "Normal" state of the toolbar item value (as indicated by the digit appended to the file name); the image on the page updates according to the current state of the toolbar item value.
3. Copy the image files to the following directory on your OEP Web server: *YourOEPwebsite*/ucf/data/custom/*yourCompany* (Onyx recommends creating a company-specific subdirectory in the custom directory).

Adding buttons and other toolbar items to toolbars

You can add the following types of toolbar items to a toolbar:

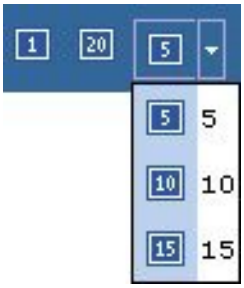
- **Button** (image with one value; only one value is allowed)
- **MultiState** (images representing multiple values available for cycling through; typically used for yes and no or on and off states)
- **Action Menu** (images and/or text representing multiple values available in a drop-down list)
- **Expand/Contract** (predefined images representing the two default values; only two values are allowed)

Newly added toolbar items are placed in the right-most position on the toolbar (see Moving toolbar items on toolbars to reposition toolbar items). To view a sample configuration including new toolbar items, see Adding toolbar buttons.

Note: You can add functionality to **Expand/Contract** toolbar items using Dynamic Forms Designer; however, the standard expand/contract functionality is not currently available.

Graphic illustrating toolbar items

The following graphic illustrates toolbar items of the following types, respectively: **Button**, **MultiState**, and **Action Menu**.



To add a toolbar item to a toolbar:

1. Add the custom toolbar.
2. Create images for each value in the toolbar item and place the image files in the appropriate UCW directory.
3. Click for the current page to enter design mode and then click on the UCW toolbar to display Navigation Designer.
4. From the **Control** drop-down list, select **Toolbar Controls** to display the existing toolbars for the current page.
5. In the Items pane, click the desired toolbar.
6. Click to add a toolbar item to the selected toolbar.
7. Select the desired toolbar item type from the **Type** text box and type the desired unique identifier in the **ID** text box, using alphanumeric characters and underscores only.

   Additional properties are displayed when you select the **Action Menu** toolbar item type.
8. If you selected the **Action Menu** toolbar item type, configure the additional properties.

   Additional properties displayed for **Action Menu** toolbar items

Property	Description
Tooltip	Text briefly displayed when the user points to the arrow on the closed drop-down list. Example: "Click to display xyz options." Also identifies the item in Navigation Designer.
Display Mode	Specifies the UI elements (images and/or text) displayed for the selected (active) value when the drop-down list is closed. UI elements include images and text.
List	Specifies the UI elements (images and/or text) displayed within the drop-down list.

Property	Description
Mode	

9. For each value you want to add to this toolbar item, click  and then configure the toolbar item value's properties.

   Properties for toolbar item values

Property	Description
ID	Unique identifier of the toolbar item value. Use alphanumeric characters and underscores only. Once set, the ID cannot be changed.
Value	Specifies the data value returned to the toolbar's Item Click event when an OEP user selects this toolbar item value. For example, if this toolbar item value is the "B" option in an Action Menu toolbar item (drop-down list) and you define the value as "2," then user selection of this toolbar item value ("B") returns the data value "2" to the toolbar's Item Click event. (This text box is not available for the Buttontoolbar item value.)
Tooltip	Text briefly displayed when the user points to the UI element representing the toolbar item value. Also identifies the value in Navigation Designer if the Caption text box is empty.
Caption	Text identifying the value on the page and in Navigation Designer. (This property is available for values in Action Menu toolbar items only.)
Image	Specifies the file name and path of the image representing the "Normal" state (digit 0) of the toolbar item value. (The image on the page updates according to the current state of the toolbar item value.) Image files for custom toolbar item values must be stored in the displayed directory *(YourOEPwebsite*/ucf/data/custom). Onyx recommends creating a company-specific subdirectory in the custom directory. Image files for default toolbar item values are stored in the *YourOEPwebsite*/images/icons directory.

   When you click  to add values to an **Expand/Contract** toolbar item, both default values and their corresponding predefined images are automatically added.

10. To reposition the toolbar item on the toolbar, drag the toolbar item's handle () to the desired position on the toolbar.

    The highest position in Navigation Designer translates to the left-most position on the toolbar.

11. Click  to confirm changes and close Navigation Designer.

    Changes made to toolbars via Navigation Designer are not immediately visible on the page. To view these changes, either exit and re-enter design mode or edit a toolbar property so changes

are detected. To edit a property, right-click the toolbar control's handle (), select **Properties**, change any property (for example, add and then remove a space), and then click to apply and view the changes.

12. Click on the UCW toolbar to save all configurations and exit design mode.

Adding functionality to toolbar items

You can add functionality to toolbar items by configuring action statements for individual values in the toolbar item. Examples of available functionality include navigating to other OEP pages and opening browser windows.

Note: You can add functionality to **Expand/Contract** toolbar items using Dynamic Forms Designer; however, the standard expand/contract functionality is not currently available.

Note: This procedure uses Dynamic Forms Designer. For more information on this window, see the Configuring page behavior book. To see a sample UI configuration demonstrating the procedure for adding functionality to toolbar items, see Adding toolbar buttons and other items.

To add functionality to a toolbar item value:

1. If required, click for the current page to enter design mode.
2. Note the identifiers for the toolbar, toolbar item, and toolbar item value, using the instructions in the following table.

Element	Identifier	Example	To view this identifier...
toolbar (custom)	**Admin ID**	CustomXToolbar	Right-click the toolbar toolbox control's handle () and then select **Properties**.
toolbar (default)	toolbar group	Top_Toolbar	Click on the UCW toolbar to display Navigation Designer, and then locate the toolbar group containing the toolbar in the left pane.
toolbar item	**ID**	toolbarItem1	Click on the UCW toolbar to display Navigation Designer, and then select the toolbar item in the left pane.
toolbar item value	**Value**	1	Click on the UCW toolbar to display Navigation Designer, and then select the toolbar item value in the left pane.

3. Click  on the UCW toolbar to display Dynamic Forms Designer.
4. Select the click event for the toolbar:
   - In the Events pane, expand **Control** and then expand the related toolbar.

     In this window, custom toolbars are identified by their respective Admin IDs; default toolbars are identified by toolbar group (such as "Top_Toolbar").
   - Select **Item Click (Item ID) (Item Value)** under the toolbar.
5. Click  to add a statement.
6. In the **Description** text box at the top of the Action Designer pane, type a description. For example, "Configure toolbar item to add a customer."
7. Add the following conditions to specify the toolbar item and toolbar item value:
   a. In the **Object** drop-down list located in the Conditions section, expand **Event Arguments** and then select **Item ID**.
   b. From the **Operator** drop-down list, select **Equal**; in the **Value** text box, type the identifier noted for the toolbar item (**ID** property).
   c. Click  in the Summary section to add the condition to the action statement.
   d. The Summary section lists the added condition as follows:
   "Event Arguments" (Item ID) is equal to [the identifier for the toolbar item]
   e. In the **Object** drop-down list located in the Conditions section, expand **Event Arguments** and then select **Item Value**.
   f. From the **Operator** drop-down list, select **Equal**; in the **Value** text box, type the identifier noted for the toolbar item value (**Value** property).
   g. Click  in the Summary section to add the condition to the action statement.

   The Summary section lists the added condition as follows:
   "Event Arguments" (Item Value) is equal to [the identifier for the toolbar item value]
8. From the **Object (Action)** drop-down list in the Action section, select the desired action and then fill in the **Source** and **Value** text boxes. For an example, see Adding toolbar buttons and other items.
9. Click  to confirm changes and then close Dynamic Forms Designer.
10. Click  on the UCW toolbar to save all configurations and exit design mode.

Modifying toolbar item values

You can modify button images and other properties of toolbar item values.

To modify a toolbar item value:

1. Click  for the current page to enter design mode and then click  on the UCW toolbar to display Navigation Designer.
2. From the **Control** drop-down list, select **Toolbar Controls** to display the existing toolbars for the current page.
3. In the Items pane, expand the related toolbar, expand the related toolbar item, click the toolbar item value to display its properties, and then change the properties as desired.
4. Click  to confirm changes and close Navigation Designer.
5. Click  on the UCW toolbar to save all configurations and exit design mode.

Moving toolbar items and toolbar item values

You can reposition toolbar items on the toolbar and reorder toolbar item values within a toolbar item (MultiState and Action Menu types). For example, you can move the second toolbar item from the left to display as the first toolbar item from the left, or you can reorder values so that the toolbar item value "B" is displayed before toolbar item value "A".

To move a toolbar item on a toolbar:

1. Click  for the current page to enter design mode and then click  on the UCW toolbar to display Navigation Designer.
2. From the **Control** drop-down list, select **Toolbar Controls** to display the existing toolbars for the current page.
3. In the Items pane, drag the toolbar item's handle () to the desired position on the toolbar.

   The highest position in Navigation Designer translates to the left-most position on the toolbar.
4. Click  to confirm changes and close Navigation Designer.
5. Click  on the UCW toolbar to save all configurations and exit design mode.

To move a toolbar item value within a toolbar item:

1. Click  for the current page to enter design mode and then click  on the UCW toolbar to display Navigation Designer.
2. From the Control drop-down list, select Toolbar Controls to display the existing toolbars for the current page.
3. In the Items pane, expand the toolbar item to display its toolbar item values.
4. Drag the toolbar item value's handle ( ) to the desired position within the toolbar item.

The highest position in Navigation Designer translates to the first position within the toolbar item.

5. Click to confirm changes and close Navigation Designer.
6. Click on the UCW toolbar to save all configurations and exit design mode.

Hiding toolbar items on toolbars

You can hide toolbar items on toolbars from users who log on to OEP with the selected profile. You can also hide toolbar items based on configured conditions. For more information on conditions, see the Configuring page behavior book.

Because this procedure uses the Load event, you must reload the page after completing the procedure to see the changes.

To hide a toolbar item on a toolbar:

1. If required, click for the current page to enter design mode.
2. Note the identifiers for the toolbar and toolbar item using the instructions in the following table.

Element	Identifier	Example	To view this identifier...
toolbar (custom)	**Admin ID**	CustomXToolbar	Right-click the toolbar toolbox control's handle ( ) and then select **Properties**.
toolbar (default)	toolbar group	Top_Toolbar	Click on the UCW toolbar to display Navigation Designer, and then locate the toolbar group containing the toolbar in the left pane.
toolbar item	**ID**	toolbarItem1	Click on the UCW toolbar to display Navigation Designer, and then select the toolbar item in the left pane.

3. Click on the UCW toolbar to display Dynamic Forms Designer.
4. In the Events pane, expand **Page** and then select **Load**.
5. Click to add an action statement.
6. In the **Description** text box at the top of the Action Designer pane, type a description. For example, "Hide toolbarItem1."
7. In the **Object (Action)** drop-down list located in the Action section, expand **Controls**, expand the related toolbar, and then select **Hide Item**.

In this window, custom toolbars are identified by their respective Admin IDs; default toolbars are identified by toolbar group (such as "Top_Toolbar").

The **Object (Action)** text box now displays the selected toolbar and action (example: **Top_ Toolbar (Hide Item)**).

8. From the **Source** drop-down list, select **User Defined**; from the **Value** drop-down list, select the identifier of the related toolbar item (such as "toolbarItem1").
9. Click to confirm changes and then close Dynamic Forms Designer.
10. Click on the UCW toolbar to save all configurations and exit design mode.

Deleting toolbar items from toolbars

You can delete custom toolbar items from toolbars. You can also delete values from custom toolbar items (Action Menu and MultiState types). Default toolbar items cannot be deleted, but you can hide toolbar items from users who log on to OEP with the selected profile.

Note: Changes made to toolbars via Navigation Designer are not immediately visible on the page. To view these changes, either exit and re-enter design mode or edit a toolbar property so changes are detected. To edit a property, right-click the toolbar control's handle (), select Properties, change any property (for example, add and then remove a space), and then click to apply and view the changes.

To delete a toolbar item from a toolbar:

1. Click for the current page to enter design mode and then click on the UCW toolbar to display Navigation Designer.
2. From the **Control** drop-down list, select **Toolbar Controls** to display the existing toolbars for the current page.
3. In the Items pane, expand the related toolbar to display its toolbar items.
4. Click the desired toolbar item and then click to delete it.
5. Click to confirm changes and close Navigation Designer.
6. Click on the UCW toolbar to save all configurations and exit design mode.

To delete a toolbar item value from a toolbar item:

1. Click for the current page to enter design mode and then click on the UCW toolbar to display Navigation Designer.
2. From the **Control** drop-down list, select **Toolbar Controls** to display the existing toolbars for the current page.

3. In the Items pane, expand the related toolbar to display its toolbar items and then expand the related toolbar item to display its toolbar item values.
4. Click the desired toolbar item value and then click to delete it.
5. Click to confirm changes and close Navigation Designer.
6. Click on the UCW toolbar to save all configurations and exit design mode.

Hiding toolbars

You can hide custom toolbars from users who log on to OEP with the selected profile. Default toolbars cannot be hidden or deleted. You can also hide toolbars based on configured conditions. For more information on conditions, see the Configuring page behavior book.

Note: You can also configure toolbar items to hide and show UI elements (such as toolbar items or tabs). To view an example configuration that hides and shows a tab, see Adding toolbar buttons and other items.

Because this procedure uses the Load event, you must reload the page after completing the procedure to see the changes.

To hide a toolbar:

1. If required, click for the current page to enter design mode.
2. Note the toolbar identifier using the instructions in the following table.

Element	Identifier	Example	To view this identifier...
toolbar (custom)	**Admin ID**	CustomXToolbar	Right-click the toolbar toolbox control's handle ( ) and then select **Properties**.
toolbar (default)	toolbar group	Top_Toolbar	Click on the UCW toolbar to display Navigation Designer, and then locate the toolbar group containing the toolbar in the left pane.

3. Click on the UCW toolbar to display Dynamic Forms Designer.
4. In the Events pane, expand **Page** and then select **Load**.
5. Click to add an action statement.
6. In the **Description** text box at the top of the Action Designer pane, type a description. For example: "Hide custom toolbar."
7. In the **Object (Action)** drop-down list located in the Action section, expand **Controls**, expand the related toolbar, and then select **Hide**.

In this window, custom toolbars are identified by their respective Admin IDs; default toolbars are identified by toolbar group (such as "Top_Toolbar").

8. Click to confirm changes and then close Dynamic Forms Designer.
9. Click on the UCW toolbar to save all configurations and exit design mode.

Deleting toolbars

You can delete custom toolbars or just remove them from the page. Default toolbars cannot be hidden or deleted.

To remove a custom toolbar (toolbox control) from the page:

- Right-click the toolbar toolbox control's handle ( ) and then select Delete.

To delete a custom toolbar:

1. Click for the current page to enter design mode and then click on the UCW toolbar to display Navigation Designer.
2. From the **Control** drop-down list, select **Toolbar Controls** to display the existing toolbars for the current page.
3. In the Items pane, click the desired toolbar.
4. Click to delete the selected toolbar.
5. Click to confirm changes and close Navigation Designer.
6. To remove the custom toolbar (toolbox control) from the page, right-click the toolbar toolbox control's handle ( ) and then select **Delete**.
7. Click on the UCW toolbar to save all configurations and exit design mode.

Configuring Tabs

Many OEP pages contain a set of tabs that enable OEP users to view different sets of data that pertain to the displayed page or record. These tabs can appear at the top of the page, the bottom of the page, or both.

The following graphic shows custom tabs on the company edit page. To learn how to add custom tabs like those depicted here, see the sample UI configuration Creating a tab.

What do you want to do?

- Add a tab to a page
- Rename a tab
- Move a tab
- Hide a tab
- Delete a tab

Procedures

Adding tabs to pages

You can add tabs to pages. To view a sample configuration for a new tab, see Creating a tab. For information on arranging UI elements within tabs, see the Configuring layout book.

Note: If you suspect that a custom tab was added previously even though you don't see it on the page, or if you don't see a default tab, you can click on the UCW toolbar to show all hidden UI elements. Tabs can be hidden dynamically (through Dynamic Forms Designer) or otherwise.

To add a tab to the current page:

1. Click for the current page to enter design mode and then click on the UCW toolbar to display Navigation Designer.
2. From the **Control** drop-down list, select **Tab Controls** to display the existing groups of tabs for the current page.
3. Specify the insertion point for the new tab:
   - To place it last (right-most tab on the page), click the desired group of tabs (such as "Powerpage Individual Top Tabs") in the Items pane.
   - To choose another position for the new tab, expand the desired group of tabs (such as "Powerpage Individual Top Tabs") in the Items pane and then click the tab before the desired insertion point.
3. Click to add the tab.
4. In the **ID** text box, type a unique identifier using alphanumeric characters and underscores only.
5. In the **Caption** text box, type the desired caption (name).
6. Click to confirm changes and close Navigation Designer.
7. Click on the UCW toolbar to save all configurations and exit design mode.

Renaming tabs

You can rename tabs by modifying the tab captions. For information on arranging UI elements within tabs, see the Configuring layout book.

To rename a tab (modify a tab caption):

1. Click  for the current page to enter design mode and then click  on the UCW toolbar to display Navigation Designer.
2. From the **Control** drop-down list, select **Tab Controls** to display the existing groups of tabs for the current page.
3. In the Items pane, expand the related tab group and then click the desired tab to display its properties in the right pane.
4. In the **Caption** text box, type the desired tab name.
5. Click  to confirm changes and close Navigation Designer.
6. Click  on the UCW toolbar to save all configurations and exit design mode.

Moving tabs

You can reposition tabs on the page. For example, you can move a tab from the right-most position on the page to the left-most position on the page.

To move a tab:

1. Click  for the current page to enter design mode and then click  on the UCW toolbar to display Navigation Designer.
2. From the **Control** drop-down list, select **Tab Controls** to display the existing groups of tabs for the current page.
3. In the Items pane, expand the related tab group and then drag the tab's handle (▪) to the desired position on the page.

The highest position in Navigation Designer translates to the left-most tab position on the page.

4. Click  to confirm changes and close Navigation Designer.
5. Click  on the UCW toolbar to save all configurations and exit design mode.

Hiding tabs

You can hide tabs from users who log on to OEP with the selected profile. For details on hiding tabs and other UI elements dynamically, based on certain conditions, see Configuring page behavior.

Because this procedure uses the Load event, you must reload the page after completing the procedure to see the changes.

A bottom tab is always displayed on the company PowerPage and on the individual PowerPage. If you hide all bottom tabs on the page, the default tab defined in User Preferences is displayed. For

example, let's say you hide all bottom tabs on the company PowerPage. The solitary displayed bottom tab depends on the OEP user's defined User Preferences. If the user has selected the Sales tab as the default tab for the company PowerPage, the Sales tab is displayed.

To create the appearance of no bottom tabs on the company or individual PowerPage, add a blank bottom tab and hide all other bottom tabs on the page. Displaying only the custom blank tab ensures that all users for the selected profile see a blank area at the bottom of the page, regardless of their defined User Preferences.

To hide a tab:

1. Click  for the current page to enter design mode and then click  on the UCW toolbar to display Dynamic Forms Designer.
2. In the Events pane, expand **Page** and then select **Load**.
3. Click  to add an action statement.
4. In the **Description** text box at the top of the Action Designer pane, type a description. For example, "Hide the ABC custom tab."
5. In the **Object (Action)** drop-down list located in the Action section, expand **Control**, expand the tab group (such as **Top_Tab**), and then select **Hide Item**.
   - The **Object (Action)** text box now displays the tab group and the selected action (example: "Top_Tab (Hide Item)").
6. From the **Source** drop-down list, select **User Defined**; from the **Value** drop-down list, select the caption (name) of the desired tab.
7. Click  to confirm changes and then close Dynamic Forms Designer.
8. Click  on the UCW toolbar to save all configurations and exit design mode.

Deleting tabs

You can delete custom tabs from the page. Default tabs cannot be deleted, but you can hide tabs from users who log on to OEP with the selected profile.

To delete a tab:

1. Click  for the current page to enter design mode and then click  on the UCW toolbar to display Navigation Designer.
2. From the **Control** drop-down list, select **Tab Controls** to display the existing groups of tabs for the current page.
3. In the Items pane, expand the related tab group to display the tabs, and then click the desired tab.
4. Click  to delete the selected tab.

5. Click  to confirm changes and close Navigation Designer.
6. Click  on the UCW toolbar to save all configurations and exit design mode.

Configuring Page Behavior

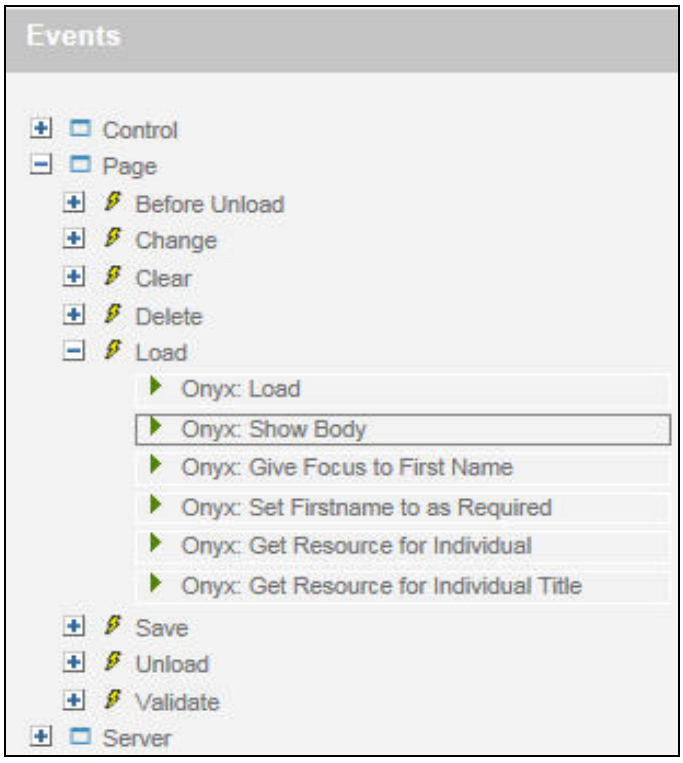
You can change the behavior of UCW-enabled areas (client-side code) by configuring action statements in relation to events for selected objects. Action statements are driven by specific events, which are initiated by objects.

For example, you can configure statements that change page behavior by dynamically highlighting UI controls, setting control values, hiding or showing UI elements, adding functionality to custom toolbar buttons, and displaying custom validation messages, to name a few types of configurations.

Configurations in Dynamic Forms Designer apply to the selected page (or to the main application frame) as identified at the top of the window.

The Events pane in Dynamic Forms Designer identifies objects, events, and statements. Custom statements (and objects and events that contain them) are indicated by boldface type. Deactivated statements are indicated by italicized type.

The following graphic illustrates statements configured for the Load event on the company edit page.



The following table describes the graphics displayed in the Events pane.

Graphic	Item	Description
	Object	A category of events: Control, Page, or Server. (Control contains objects for UI elements, which each contain events, when the element is configurable.) The Control object lists Change events related to UI elements. The Page object lists events related to the selected UCW-enabled area, such as Load, Save, and Validate. The Server object is used for configuring page appearance (not action statements).
	Event	A significant occurrence related to an object, such as the Load event for the Page object. Events are listed in alphabetical order (not sequential order).
, , or	Action statement	A configured conditional ( ) or unconditional ( ) response to an event, such as setting a default value for a control when the selected UCW-enabled area is loaded (Load event for the Page object). (indicates an action statement in Advanced View.)

For more information on the objects, events, and statements seen in the Events pane of Dynamic Forms Designer, see Dynamic Forms Designer reference.

To launch Dynamic Forms Designer:

- Click to enter design mode and then click on the UCW toolbar.

What would you like to do?

- View events and action statements
- View a list of sample UI configuration procedures
- Add a statement
- Modify a statement
- Clone a statement
- Move a statement
- Deactivate a statement
- Protect a statement
- Delete a statement
- Add a condition to an action statement
- Delete a condition from an action statement
- Learn about Advanced View

Viewing events and action statements

You can view action statements for a selected event.

To view action statements for a selected event:

1. Click  to enter design mode and then click  on the UCW toolbar to display Dynamic Forms Designer.
2. Expand the desired object to display its list of events, and then expand the desired event to display the action statements for the event.
3. To view the configuration of an action statement, click the statement.

Viewing sample UI configuration procedures for page behavior

Click the desired link below to view a sample UI configuration procedure for page behavior.

- Setting default values
- Setting controls as required
- Highlighting controls
- Changing pages dynamically
- Advanced configuration examples

Adding Statements

You can add conditional or unconditional action statements using the Action Designer pane in Dynamic Forms Designer. For example, you can hide a Header Bar button when a value on the displayed record meets specified conditions. Statements configured in the standard view of Action Designer are checked for validity.

Newly added statements are appended to the list of statements for the currently selected event (see Moving statements to reposition statements).

To add a statement:

1. Click  to enter design mode and then click  on the UCW toolbar to display Dynamic Forms Designer.

   The context (main application frame or selected page) is identified at the top of the window.

2. Select the desired event from the Events pane and then click  to display the Action Designer pane.
3. To hide the Events pane, click  .

4. In the Description text box at the top of the Action Designer pane, type a description of the statement. (Descriptions are optional for the standard view of Action Designer.)
5. To create an action statement using the standard view of Action Designer, use the following steps.

   a. If desired, add one or more conditions under which to execute the configured action. For more information on available conditions, see Actions and conditions.

   b. In the Action section of the Action Designer pane, select the object and action ( ) from the Object (Action) drop-down list and fill out the displayed Source and Value text boxes. For more information on available actions, see Actions and conditions.

   c. To store the result of this statement as a page variable, select the Save Result? check box (located in the Action section of the Action Designer pane) and then type a name for the page variable in the Variable text box.

   d. Previously stored page variables are listed in either the Page Variable source (for action arguments) or the Variables object (for conditions).

   e. To prevent execution of subsequent statements within the current event if condition(s) are met for this statement, select the Stop on Execute? check box in the Action section of the Action Designer pane.

   You can use Stop on Execute? to ensure that the result of this statement is the return value for the current event. For example, you could prevent a save during the Validate event. For more information on the Stop on Execute? check box, see Actions and conditions.

6. To create a code-based action statement, click  to enable Advanced View and then enter code in the Code section, following the coding guidelines.
7. Click  to confirm changes and then close Dynamic Forms Designer.
8. Click  on the UCW toolbar to save all configurations and exit design mode.

Modifying Statements

You can modify action statements. For example, you can change the statement description, add or delete conditions, choose another action, and toggle Advanced View with the standard view of Action Designer. Statements configured in the standard view of Action Designer are checked for validity. For information on available actions and conditions, see Actions and conditions.

To modify a statement:

1. Click  to enter design mode and then click  on the UCW toolbar to display Dynamic Forms Designer.
2. Expand the desired object and event in the Events pane to display the event's statements and then select the desired statement to display its configuration in the Action Designer pane.

3. To modify the description, edit the text in the **Description** text box.
4. If the statement is displayed in Advanced View, change the code in the Code section as desired, following the coding guidelines.
5. If the statement is in the standard view of Action Designer, use the following steps:
   a. Add conditions or delete conditions as desired.
   b. To choose another action, select the object and action ( ) from the **Object (Action)** drop-down list and fill out the displayed **Source** and **Value** text boxes.
   c. To toggle views (Advanced View, then standard view), click  and then click  .

      Statements configured in Advanced View can no longer be viewed in the standard view of Action Designer.
6. Click  to confirm changes and then close Dynamic Forms Designer.
7. Click  on the UCW toolbar to save all configurations and exit design mode.

Cloning Statements

Cloning (copying) statements can help avoid repetitive configurations.

You might want to clone a statement when you want to create a similar statement with the same complex conditions or complex action configuration. For example, you might want to clone an existing validation statement that displays a custom message if a required control is empty. You can clone this statement several times and then change just the required control in each copy of the statement. In this example, cloning would help you avoid recreating the same custom message configuration multiple times.

To clone a statement:

1. Click  to enter design mode and then click  on the UCW toolbar to display Dynamic Forms Designer.
2. Select the desired statement in the Events pane and then click  .
3. Edit the text in the **Description** text box at the top of the Action Designer pane.
4. Move the statement to the desired location.
5. Modify the statement as desired.
6. Click  to confirm changes and then close Dynamic Forms Designer.
7. Click  on the UCW toolbar to save all configurations and exit design mode.

Moving Statements

You can move action statements to another event or to another position within the sequence of statements for the selected event.

Moving statements can be useful when the results of a statement (such as setting a variable) are referenced by another statement. In this situation, the statement referencing the variable should be moved later in sequence than the statement setting the variable.

To move (resequence) a statement:

1. Click  to enter design mode and then click  on the UCW toolbar to display Dynamic Forms Designer.
2. Select the desired statement in the Events pane.
3. To reposition the statement within the sequence of statements for the selected event, click  or  .
4. To move the statement to another event (or within the current sequence of statements, above or below another statement), drag the statement's handle (or or ) to the destination event ().

   Statements with unconfirmed changes cannot be dragged. To confirm changes, click  .
5. Close Dynamic Forms Designer.
6. Click  on the UCW toolbar to save all configurations and exit design mode.

Deactivating Statements

You can deactivate a statement if you want to retain the overall statement configuration without executing it.

Deactivating default Onyx statements can be useful when you want to use custom statements instead.

Note: Onyx recommends deactivating default Onyx statements instead of deleting them as deleted statements cannot be retrieved.

To deactivate a statement:

1. Click  to enter design mode and then click  on the UCW toolbar to display Dynamic Forms Designer.
2. Select the desired statement in the Events pane to display its configuration in the Action Designer pane.

3. Clear the **Active** check box to the right of the Description text box.
4. Click  to confirm changes.

   The statement description listed in the Events pane is formatted with italics to indicate its deactivated status.
5. Close Dynamic Forms Designer.
6. Click  on the UCW toolbar to save all configurations and exit design mode.

Protecting Statements

You can protect statements from accidental modification by enabling display of a confirmation message box whenever anyone attempts to update the statements.

To protect a statement:

1. Click  to enter design mode and then click  on the UCW toolbar to display Dynamic Forms Designer.
2. Select the desired statement in the Events pane to display its configuration in the Action Designer pane.
3. Select the **Confirm Update** check box to the right of the Description text box.
4. Click  to confirm changes and then close Dynamic Forms Designer.
5. Click  on the UCW toolbar to save all configurations and exit design mode.

Deleting Statements

You can delete statements to remove them entirely from Dynamic Forms Designer. Deleted statements cannot be recovered; they must be recreated.

 **Note:** Onyx recommends deactivating default Onyx statements instead of deleting them as deleted statements cannot be retrieved.

To delete a statement:

1. Click  to enter design mode and then click  on the UCW toolbar to display Dynamic Forms Designer.
2. Select the desired statement in the Events pane to display its configuration in the Action Designer pane.
3. Click  and then confirm the operation.

4. Close Dynamic Forms Designer.
5. Click  on the UCW toolbar to save all configurations and exit design mode.

Adding Conditions

You can add conditions to action statements to describe requirements for executing the selected action. For example, you can create a statement highlighting the Subtype control if the record is an active lead. (See the sample UI configuration Highlighting controls for more information.)

To add a condition to an action statement:

1. Click  to enter design mode and then click  on the UCW toolbar to display Dynamic Forms Designer.
2. Select the desired statement in the Events pane to display its configuration in the Action Designer pane.
3. If the statement is in the standard view of Action Designer, use the following steps:
   a. In the Conditions section, select an object from the **Object** drop-down list. For more information on available conditions, see Actions and conditions.
   b. From the **Operator** drop-down list, select an operator; from the Value drop-down list (if applicable), select a value.
   c. Indicate the insertion point of this condition by clicking the desired  in the Summary section. (Multiple insertion points exist after you add the first condition to the statement.)

   If you insert subsequent conditions below the "OR" operator, only one of the conditions (or sets of conditions) must be satisfied for the action to occur.

   If you insert subsequent conditions above the "OR" operator, an implicit "AND" operator exists; all conditions must be satisfied for the action to occur.
4. If the statement is in Advanced View, insert the code that represents the condition into the correct position within the code.
5. Click  to confirm changes and then close Dynamic Forms Designer.
6. Click  on the UCW toolbar to save all configurations and exit design mode.

Deleting Conditions

You can delete conditions to remove them entirely from Dynamic Forms Designer. For example, you can delete the conditions for a statement that highlights the Subtype control. Deleted conditions cannot be recovered; they must be recreated.

To delete a condition within an action statement:

1. Click  to enter design mode and then click  on the UCW toolbar to display Dynamic Forms Designer.
2. Select the desired statement in the Events pane to display its configuration in the Action Designer pane.
3. If the statement is in the standard view of Action Designer, click  to the left of the desired condition (in the Summary section).
4. If the statement is in Advanced View, delete the code representing the condition.
5. Click  to confirm changes and then close Dynamic Forms Designer.
6. Click  on the UCW toolbar to save all configurations and exit design mode.

About Advanced View

You can create code-based action statements using Advanced View of Action Designer in Dynamic Forms Designer.

Whenever feasible, use the standard view of Action Designer to create and modify statements to ensure valid, upgradeable configurations. Advanced View is best reserved to create statements that integrate HTML Container toolbox controls, include multiple actions and/or complex conditions, use variables (that are not page variables), or use complex logic (such as XML parsers for new ActiveX objects) outside the scope of the actions and objects available from the standard view.

Unlike the standard view of Action Designer, Advanced View is capable of bundling multiple actions and using objects not available in the standard view of Dynamic Forms Designer, such as custom styles. Advanced View can also be used to filter domain data by profile.

For more information on objects, events, and statements, see Dynamic Forms Designer reference.

Note: When you confirm changes to a statement during Advanced View, the standard view of Action Designer is no longer available for that statement. To view sample configurations for statements configured in Advanced View, see the Advanced configuration examples book.

Skill set required

To configure statements using Advanced View in Dynamic Forms Designer, you should be familiar with writing, editing, and debugging JavaScript-formatted scripts and style sheets.

Note: See the Advanced configuration examples book for examples of configurations that use Advanced View in Dynamic Forms Designer.

Coding guidelines for Advanced View

Dynamic Forms Designer does not validate the code entered in Advanced View. Consider validating your code using a tool such as Microsoft Visual Studio.

When writing code in Advanced View, take care to ensure that:

- Your code complies with JavaScript coding standards.
- Your code is supported by Internet Explorer.
- Your code accesses page objects (such as ucf.page.powerpageIndividual) by invoking the action broker only (no direct access); returns a valid UcfActionReturn object instance when using the action broker.
- Your code references external Web sites (those located outside YourOEPwebsite/ucf/data) using the HTML Container toolbox control.

Calling common functions defined on the main application frame

Functions entered in <SCRIPT> blocks on the main application frame are available to all OEP pages. However, the pages cannot call such functions directly. For example, to call a custom function named myCustomFunction() that is defined for the main application frame using the ASP Editor, use the following code:

```
moUcfPageContext.getApplicationWindow().myCustomFunction()
```

OEP instantiates the UcfPageContext object for the main application frame (and other UCW-enabled areas) to support Dynamic Forms actions. Because the main application frame always exists when the OEP client is running, its context data is available through the UcfPageContext object. The getApplicationWindow() function on the object references the main application frame so that myCustomFunction() can be accessed.

Procedures

You can enable Advanced View for an action statement and you can display the list of actions that are available during the standard view of Action Designer. Displaying this list of actions enables you to select an action to insert.

When you select an action to insert, the code related to the action is inserted at the position of the cursor. If you select (highlight) a portion of code before displaying the list of actions, the code related to the selected action replaces the selected portion of the code.

To enable Advanced View for the selected action statement:

- Click  .

To display the list of actions during Advanced View:

1. Position the cursor in the Code section.
2. Press CTRL+K.

Configuring Page Appearance (ASP Editor)

Using UCW, you can add and modify code fragments to server-side ASP pages for UCW-enabled OEP pages (and for the main application frame) without editing the ASP pages themselves. Code fragments are inserted outside of the BODY element of the ASP page.

Use code fragments to:

- Add stylesheet references
- Add inline ASP script
- Add client-side script files
- Modify inline styles or embedded style sheets
- Include files, such as style sheets, JavaScript, CSS, and ASP files

Note: Dynamic Forms Designer is also used for configuring page behavior.

Skill set required

To configure page appearance using ASP Editor, you should be familiar with writing, editing, and debugging scripts and style sheets.

Coding guidelines for ASP Editor

When writing code in ASP Editor, take care to ensure that:

- Any scripts you include adhere to the coding requirements for Advanced View.
- Your code complies with JavaScript or VB Script standards. Referenced JavaScript files must declare the language using script tags; ASP Editor excepts VB Script unless you indicate otherwise. (Automatically installed JavaScript files can be found in *YourOEPwebsite*/OnyxCommon and in *YourOEPwebsite*/Images. Place your custom JavaScript files in your company-specific subdirectory under *YourOEPwebsite*/ucf/data/custom/.)
- Your code is supported by Internet Explorer.

What would you like to do?

- Add a code fragment to an ASP page
- Modify an existing code fragment
- View the list of skills required to add or modify code fragments
- View a sample configuration for including custom CSS/JS files

Adding code fragments to ASP pages

You can add code fragments to ASP pages. Code fragments must comply with HTML and ASP standards. Client-side code (such as code within <SCRIPT> blocks) must comply with other coding guidelines as well (see coding guidelines for Advanced View).

To add a code fragment to an ASP page:

1. Display the related OEP page (or the main application frame), click  to enter design mode, and then click  on the UCW toolbar to display Dynamic Forms Designer.

   The context (main application frame or selected page) is identified at the top of Dynamic Forms Designer.

2. In the Events pane, expand **Server** and then expand **Body**.

   The Before and After events are displayed in the Events pane. These events indicate available insertion points relative to the BODY element of the related ASP page.

3. Specify the fragment's insertion point by selecting either **Before** or **After**.
4. Click  to display ASP Editor.
5. Type a description of the code fragment in the Description text box.
6. Enter the desired code in the ASP/HTML section.
7. To protect this code fragment from accidental updates, select the **Confirm Update** check box to the right of the **Description** text box.
8. Click  to confirm changes and then close Dynamic Forms Designer.
9. Click  on the UCW toolbar to save all configurations and exit design mode.

Modifying existing code fragments

You can modify existing code fragments. Code fragments must comply with HTML and ASP standards. Client-side code (such as code within <SCRIPT> blocks) must comply with other coding guidelines as well (see coding guidelines for Advanced View).

To modify a code fragment:

1. Display the related OEP page (or the main application frame), click  to enter design mode, and then click  on the UCW toolbar to display Dynamic Forms Designer.

   The context (main application frame or selected page) is identified at the top of Dynamic Forms Designer.

2. In the Events pane, expand **Server**, expand **Body**, and then expand the appropriate event (**Before** or **After**) to display the desired code fragment description.
3. Select the fragment description to display the code fragment in the ASP/HTML section on the right of the window; change configuration as desired.

4. To temporarily deactivate the code fragment, clear the **Active** check box to the right of the Description text box. (You can reactivate the code fragment later by re-selecting the **Active** check box.)
5. Click  to confirm changes and then close Dynamic Forms Designer.
6. Click  on the UCW toolbar to save all configurations and exit design mode.

Including custom files

You can include custom files to change the appearance of UCW-enabled areas. For a configuration example using the ASP Editor, see Applying a custom CSS to a page.

To include a custom file for a UCW-enabled area:

1. Copy the custom file to ... Onyx\EmployeePortal\stylesheet\
2. Display the related OEP page (or the main application frame), click  to enter design mode, and then click  on the UCW toolbar to display Dynamic Forms Designer.

   The context (main application frame or selected page) is identified at the top of Dynamic Forms Designer.
3. In the Events pane, expand **Server** and then expand **Body**.

   The **Before** and **After** events are displayed in the Events pane. These events indicate available insertion points relative to the BODY element of the ASP page for the current context (selected page or the main application frame).
4. Select the Before event and then click  to display the ASP/HTML section on the right of the window.
5. Type a description of the code fragment in the **Description** text box above the ASP/HTML section. (Example: Include custom file for logos)
6. To include a custom CSS file: In the **ASP/HTML** section, type <link rel="stylesheet" type="text/css" href="yourfilename"> where yourfilename is the name of your custom CSS file.
7. To include a custom JavaScript file: In the **ASP/HTML** section, type <script src="yourfilename"></script> where yourfilename is the name of your custom JavaScript file.
8. To protect this code fragment from accidental updates, select the **Confirm Update** check box to the right of the Description text box.
9. Click  to confirm changes and then close Dynamic Forms Designer.
10. Click  on the UCW toolbar to save all configurations and exit design mode.

Configuring UI and Message Text

With UCW, designers can configure nearly all of the existing UI and message text that the user sees on OEP pages using the following tools:

- Navigation Designer to configure the Header Bar, Navigation Bar, toolbar and tab captions
- The Properties window in design mode to customize controls, labels, and titles
- UI Text Editor to modify the text that exists in a UI resource file, such as error messages, dialogs, and non-custom tooltips

UCW or OES?

Modifying text with UCW differs from modifying text with OES in that the changes made in UCW supersede changes made in OES. So, for example, if you use OES Reference Table Administration to define a caption for a UDF and then rename the caption in UCW, the page will display the caption that you defined via UCW, regardless of the number of times you rename the control in OES Reference Table Administration.

Another way in which the two methods differ is that modifications made using UCW can be unique to each profile you create, while those made using the OES Reference Table Administrator are global.

What would you like to do?

- Configure text in the Properties window
- Learn more about the UI Text Editor

Configuring UI Text

You can configure text directly on the current page - and view your changes immediately - when you use the Properties window to configure UI text.

Note: There are times when part of a value in the caption or text box of the Properties window is replaced by the characters "~1." For example "~1 Address" may represent "Billing Address." The "~1" is used in lieu of the word, "Billing." In modifying the caption or text, you can replace the "~1" with whatever text you choose. The change will only affect the value you replaced. It will not affect other strings that use "~1."

To configure UI text from the Properties window

1. Click  to enter design mode.
2. Right-click the text you want to modify, and select **Edit Properties**.
3. Modify the item's text in the Properties window located on the left side of the screen.
4. Click  in the Properties window to refresh the UI and view your change.
5. Click  to save your changes.

Using UI Text Editor

Some existing UI text, such as messages and dialog boxes, can only be modified with UI Text Editor. And, though it is possible to modify captions, labels, titles, and tooltips with UI Text Editor, the UI Text Editor requires you to refresh the page before you can view your changes. So it may be more convenient to modify these UI items in the Properties window while in design mode.

To launch UI Text Editor:

- Click  to enter design mode and then click  on the UCW toolbar to display UI Text Editor.

  The UI Text Editor opens in the context of the host page, and the host page defines the text that can be configured. For example, if the UI Text Editor is launched from the Incident Edit page, only text in the incident edit page is configurable.

What would you like to do?

- Add a custom UI message
- Modify a custom UI message
- Delete a custom UI message

Adding a custom UI message

Creating a new text item is a two-step process. First, you must create the text item via UI Text Editor, then you must associate the text item with an action or event via the Dynamic Forms Designer.

Create the message content

1. Open the UI Text Editor.
2. Select a parent node from the Item pane and click  to create a new text item.

UI Text ID prefixes	Definition
caption	Text identifying section titles
data	Text regarding data entry
label	Text identifying fields
msg	Text inside error messages or dialog boxes
save	Text regarding the save status of the entry
tip	(tooltip) Text briefly displayed when the user points to the button
title	Text identifying a Header Bar button
ucf	(UI Control Fields) The text in text boxes, drop down lists, tabs, toolbars, and so on
WorkbenchPageName	Text identifying the page to which you are adding UI text items

3. Assign the text an ID in the Name field. You can enter any string of alphanumeric and underscore characters. For example, customMessageText. Remember the name of the message ID.
4. In the Text field, type the desired message.
5. Click  to confirm your change and close the UI Text Editor window.
6. Click  in the page to save your changes.

Associate the message with an event

To learn how to assign your text to an event using the Dynamic Forms Designer, please read:

- Adding statements
- Modifying statements
- Setting controls as required - Saving a resource string as a page variable
- Setting controls as required - Configuring a validation error message using the saved page variable

Modifying a custom UI message

Note: There are times when part of a value is replaced by the characters "~1"in the text box of the Properties window. For example "~1 Address" may represent "Billing Address." The "~1" is used in lieu of the word, "Billing." In modifying the caption or text, you can replace the "~1" with whatever text you choose. The change will only affect the value you replaced. It will not affect other strings that use "~1."

To modify a message

1. Open the UI Text Editor.
2. Select the message ID from the parent node in the Item pane.
3. Modify the contents of the Text field as needed.
4. Click  to confirm your change and close the UI Text Editor window.
5. Click  in the page to save your changes.

Note: Changes you make via the UI Text Editor are not saved in their corresponding resource files, so make sure to use UCW to configure UI text rather than relying on the resource files.

Deleting a custom UI message

To delete a message

1. Open the UI Text Editor.
2. Select the message ID from the parent node in the Item pane.

3. Click  to delete the message.
4. Click  to confirm your change and close the UI Text Editor window.
5. Click  in the page to save your changes.

Setting Default OEP User Preferences

OEP users inherit user preference settings that are established for their OEP profile by a special user account called User Preferences Default. Users can override these settings by setting their own user preferences within OEP.

Note: It's typically better to set default OEP user preferences in your test or production environments rather than in your development environment. If you instead choose to set them in the development environment, you must then migrate the data from your development environment to your test or production environments.

There are two main types of user preferences:

- **General system preferences:** Preferences can be set for various feature areas in OEP. For instance, one preference specifies your OEP start page, and another preference specifies the default currency type for forecast and quote records. To set preferences such as these, click OEP's Preferences button and adjust your settings as desired.
- **Result list preferences:** Preferences can also be set for result lists, which enables you to define how the retrieved data (per result list) is presented to various users. For instance, you can configure the width of each column of the result list, the order in which each column appears, and whether or not the result list displays a certain column at all. To configure a result list, click the **Configure List Settings** button located at the bottom of the page, and configure the list as desired.

The User Preferences Default account

To set default user preferences, you must first enable the User Preferences Default account. You can then set default user preferences for all OEP profiles simultaneously (by logging on to OEP using the Global profile) or for each OEP profile separately (by logging on to OEP using the desired OEP profile).

The User Preferences Default account is a special user account that is designed for managing only default user preferences; it cannot be used to reset preferences that users set themselves. This means that if you, as administrator, use the User Preferences Default account to modify OEP's default preference settings for a certain UI, users that belong to the OEP profile associated with that UI will inherit only the settings that they have not already defined for themselves.

For example, if you use the User Preferences Default account to change the Startup View from Home Page to Task Manager, only the OEP users who have not already defined their own preferred Startup View will inherit this change. The next time these users launch OEP, they will see Task Manager as their start page. Users who have selected their own Startup View will continue to see that as their start page.

Enabling the User Preferences Default Account

The User Preferences Default account is one of several OEP user accounts created during OEP setup. Although this account is designed only for setting default user preferences, it is like any other user account in OEP and should be treated as such. If given sufficient permissions, it can be used to add, edit, and remove information from the database.

After OEP setup is complete, the User Preferences Default account exists but it has no permissions and it does not belong to any security roles. Until you configure it using OES Security Administration, you cannot use it to set default user preferences. In fact, until you configure it for use, it doesn't even have permission to log on to OEP.

To use the User Preferences Default account, you must grant it permissions to log on to OEP and to view the result lists for each OEP feature. The simplest way to accomplish this is to add the User Preferences Default account to the OEP.user role. After you have finished making desired changes to the default preference settings, you can either remove the User Preferences Default account from the OEP.user role, or you can disable it entirely.

After enabling the User Preferences Default account, you can set default user preferences for all profiles, and you can set default user preferences for distinct OEP profiles.

To enable the User Preferences Default account:

1. Launch OES Security Administration.
2. From the **Security Elements** pane, navigate to Groups\Administrators\Users and select the user account called **User Preferences Default**.
3. In the **Edit User** pane, select **Active** to activate this account.
4. Select either Onyx authentication or integrated Windows authentication.
5. If desired, change the password for the User Preferences Default account by typing a new value into the **Password** field. The default password is onyx.
6. Click **Save**.
7. Grant the User Preferences Default account permissions to the OEP.user role.
    a. From the **Security Elements** pane, navigate to Roles and select the role called **OEP.user**.
    b. In the **Edit Role** pane, click **Add/Remove Users**.
    c. From the **Security Elements** pane, navigate to Groups\Administrators\Users and select the user account called **User Preferences Default**.
    d. From the **Security Elements** pane, click **Add**.
    e. Click **Done**.
    f. Click **Save**.

Tip: After logging in to OEP and setting default preferences as desired, consider removing this account from the OEP.user role or deactivating it entirely.

Setting Default User Preferences (Global profile)

To set default user preferences for all OEP profiles simultaneously, log on to OEP using the User Preferences Default account and the Global profile, and then change the preferences as any other user would.

Because all OEP profiles inherit from the Global profile, modifications made to this profile are also made to all other OEP profiles. Be aware, however, that once you set a preference within a certain OEP profile, that profile no longer inherits that preference from the Global profile. To modify that preference, you must log on to OEP using that OEP profile.

If you are resetting the default user preferences in your production environment, remember that your users will receive only the settings that they have not already defined for themselves.

To configure default user preferences for all profiles:

1. If you haven't done so already, enable the User Preferences Default account so you can use that account to set default user preferences.
2. Log on to OEP using the User Preferences Default account. Use the following logon credentials:

   Login: default

   Password: *onyx* (unless you changed this password when configuring this account)

   Profile: Global
3. From OEP's main toolbar, click **Preferences** and set preferences as desired.
4. Configure each search result list as desired. See the OEP Help for details on configuring result lists.
5. Log out of OEP.
6. Stop and restart your OEP Web server, such as by using the Internet Information Services (IIS) Manager tool or by executing the IISReset command (via Command Prompt).

Setting Default User Preferences (by OEP profile)

To set default user preferences for an OEP profile, log on to OEP using the User Preferences Default account and the desired OEP profile, and then change the preferences as any other user would.

Remember that preferences set for an OEP profile override the settings that it had inherited from the Global profile.

If you are resetting the default user preferences in your production environment, remember that your users will receive only the settings that they have not already defined for themselves.

To configure default user preferences for an OEP profile:

1. If you haven't done so already, enable the User Preferences Default account so you can use that account to set default preferences.
2. Log on to OEP using the User Preferences Default account. Use the following logon credentials:

   Login: default

   Password: onyx (unless you changed this password when configuring this account)

   Profile: Select the OEP profile whose default preferences you want to set
3. From OEP's main toolbar, click **Preferences** and set preferences as desired.
4. Configure each search result list as desired. See the OEP Help for details on configuring result lists.
5. Log out of OEP.
6. Stop and restart your OEP Web server, such as by using the Internet Information Services (IIS) Manager tool or by executing the IISReset command (via Command Prompt).

Troubleshooting Configuration

This topic includes troubleshooting tips for resolving issues with UCW configurations.

Modifying the baseline UI generates an internal error (UcfException)

Problem: Modifications to the baseline UI (the UI associated with the Global profile) caused problems in one or more profile-specific UIs. Logging on to a profile-specific UI generates an internal "UcfException" error.

Issue: Profile-specific UIs inherit the appearance, behavior, and other such characteristics from the baseline UI. Because profile-specific UIs rely on certain data in the baseline UI, modifications to the baseline UI can produce unpredictable results in all profile-specific UIs.

Resolution: Exit OEP and log back on. Select the Global profile, check **Disable custom actions**, and then click **OK**. After logging on, navigate to the affected page and click **Reset Page Configurations**.

Changes to the toolbar are not displaying

Problem: Changes made to toolbars via Navigation Designer are not immediately visible on the page.

Issue: The changes were not detected.

Resolution: To view the changes, either exit and re-enter design mode or edit a toolbar property so changes are detected. To edit a property, right-click the toolbar control's handle (), select **Properties**, change any property (for example, add and then remove a space), and then click to apply and view the changes.

Duplicate toolbox controls in Dynamic Forms Designer

Problem: In Dynamic Forms Designer, duplicate toolbox controls are sometimes listed, making it difficult to ensure selection of the correct control.

Issue: An Admin ID property is shared by multiple toolbox controls on the page. This situation occurs when you retain the default properties for toolbox controls on multiple tabs of a page. Default Admin ID properties (such as "toolbox_1") are unique for toolbox controls within a single tab.

Resolution: Change the Admin ID properties to ensure unique Admin IDs for each toolbox control on the page.

Toolbox controls are not saving values (UDFs on task or incident pages)

Problem: Toolbox controls that reference UDF fields on the task or incident pages are not saving values. (Reference data is available for Dropdown and Text Reference toolbox controls only.)

Issue: The Parent ID was specified to indicate the category, but the category for UDFs on these pages is indicated by the field name (such as "incident.sales.user8").

Resolution: Leave the Parent ID property empty. Right-click the toolbar control's handle (), select Properties, delete the entry in the Parent ID property, and then click to apply the changes.

OEP logon fails, generates Internal Error -2147467259

Problem: When attempting to log on to OEP using a certain OEP profile, the logon attempt fails and OEP returns the following Internal Error. "An error occurred when transferring to the target page."

Issue: This problem occurs if all necessary UCW files that pertain to that OEP profile have not been published to your production OEP Web server.

Resolution: To resolve this problem, republish all UCW files from your development environment to your production environment.

Debugging Your Changes

UCW provides several ways for you to debug script code and ASP statements used with Dynamic Forms Designer. When working within the Advanced View, you can use shortcut keys to immediately debug the code that you are writing.

Disabling custom actions

When configuring OEP using UCW, you may implement a configuration that generates errors that prohibit the page from loading normally. If this happens, you can check the **Disable custom actions** option when logging on to OEP.

Disabling custom actions temporarily disables all script and ASP statements from executing, thereby enabling you to enter design mode, launch Dynamic Forms, and fix your code as necessary.

To disable custom actions:

- Exit OEP and log back on.
- If you're using Onyx authentication, provide your user name and password in the first logon window. Click **OK**.
- In the next logon window, select the desired profile and check **Disable custom actions**.
- Click **OK**.

Debugging using shortcut keys

When the cursor is within the Advanced View editor, several keystrokes assist debugging. Ctrl-D followed by Ctrl-I executes the code in the context of the OEP page being configured. This keystroke sequence executes code regardless of whether you selected the Disable custom actions option when you logged on the OEP.

If you have Visual Studio installed on your OEP client machine, you can add "debugger;" to the start of your Advanced View code before executing it. The debugger statement starts the Visual Studio script debugger, which lets you single-step through the script code, examine variable values, and so forth.

Compiling Your Changes

Compiling UCW configurations generates a set of ASP pages that incorporates all UI modifications that have been made to OEP via UCW.

When working in the development environment, UCW automatically compiles pages each time you open them. For instance, let's say that you've configured the incident edit page, saved your changes, and closed that page. The next time you open that page, UCW automatically compiles it before opening it. Once it is open, you can view and test all configurations that you saved to that page.

Because pages aren't compiled until you open them, it's good practice to periodically perform a manual compile to ensure that all pages are compiled. For the same reason, it's a good idea to perform a compile before publishing your changes for your test or production environment.

To compile your changes:

1. Click  .
2. In the Design Mode window, click **Yes**. UCW compiles all UCW-enabled pages simultaneously.

   When you compile changes in the baseline UI, UCW automatically compiles changes made to all profile-specific UIs, as well as the changes made to the baseline UI.
3. When the compile process is complete, click **OK**.

UCW-configured pages, once compiled, can be debugged like any ASP page.

Publish Your Changes

After you compile and test your configurations, you can publish your changes to your test or production environment. When you publish UCW-configured OEP pages, you copy them to the OEP Web server's UCF cache. Pages stored in the UCF cache are retrieved quickly and don't need to be re-compiled each time a user requests them.

To publish UCW-configured pages:

1. From the UCF cache directory of your development environment's Web server, copy the following folders:
   - **ucf/data/layers/active** (This folder contains data that is compiled dynamically at run-time, when the page is requested by the client. This includes the navigation and header bars of the main OEP application pages, and the UI text for all pages.)
   - **ucf/data/custom** (This folder contains any custom images that you may have created.)
   - ucf/data/published (This folder contains all compiled UCW-configured pages.)
2. Paste the copied folders to their corresponding location (within the UCF cache directory) of your test or production environment's Web server.
3. Stop and restart your OEP Web server, such as by using the Internet Information Services (IIS) Manager tool or by executing the IISReset command (via Command Prompt).
4. Using OES Security Administration, ensure that the Primary ID of each OEP profile in your test or production environment is identical to the Primary ID established in the development environment. Otherwise, users may have difficulty logging on to OEP.

If a Primary ID does not match, use OES Security Administration to add (to the test or production database) a new OEP profile with the correct Primary ID.

Starting Over

You can remove all UCW configurations for a given page/profile with a click of a button. The result of resetting your configurations depends on whether you're using the baseline UI or a profile-specific UI.

If you're using this UI...	...then this happens
Baseline UI (the UI associated with the Global profile)	The page is reset to the default OEP page (as it appears when first installed to your system).
Profile-specific UI	The page is reset to become identical to its corresponding page in the baseline UI.

To reset your configurations:

- Click  .

Sample UI Configurations

The sample UI configurations in this book include step-by-step instructions and graphics to demonstrate the following UCW design mode configurations.

- Configuring the Header Bar and Navigation Bar
- Creating tabs and adding custom UI elements
- Configuring controls (default values, required selections, and highlighted appearance)
- Dynamically changing pages in response to users' selections
- Advanced configuration examples using Advanced View in Dynamic Forms Designer

Adding a Header Bar Button

Using UCW, you can add buttons to the Header Bar without any programming knowledge.

This sample UI configuration demonstrates configuration of a Header Bar button that accesses the Google Web site, displayed for users with specific UI resource permissions, and highlighting of the button for a specific user.

Scenario used in this sample UI configuration

Employees who create marketing campaigns ask for quick access from OEP to the Google Web site, but others in the company do not want to see the extra button on the Header Bar. One employee, the manager, wants the custom button to be highlighted when she first launches OEP.

The following graphic illustrates the custom Header Bar button ("Analytics") on the right of the Header Bar.

Procedures in this sample UI configuration

Follow these procedures to add the Google Web site button to the Header Bar and to highlight the custom button for the manager.

- Add a Google Web site button to the Header Bar
- Highlight the Google Web site button on the Header Bar

Adding a Google Web site button to the Header Bar

You can add a button to the Header Bar that accesses the Google Web site in a separate browser window. This procedure also specifies window height and width of 400 pixels.

In this procedure, you limit display of the custom Header Bar button to users with permissions for the "Campaign Administration" UI resource.

Note: For basic instructions on configuring the Header Bar, see Configuring the Header Bar.

To add a Google Web site button to the Header Bar:

- Copy the sample images "googleOn.gif" and "googleOff.gif" from the ../TechnicalGuide/Examples/Images/Header Bar directory on the OEP product CD to the following directory on your OEP Web server: *YourOEPwebsite*/ucf/data/custom/sample

  Onyx recommends creating a company-specific subdirectory under *YourOEPwebsite*/ucf/data/custom/ for your own images.

- In Navigation Designer ( ) for the main application frame, add the Header Bar button using the property values listed in the following table (typed entries are indicated by italics; selected entries are indicated by boldface). (See Configuring the Header Bar for basic instructions on adding buttons to the Header Bar.)

Property	Value
ID	headerBarGoogle
Caption	Google
Tooltip	Go to Google.com
URL	http://www.google.com
UI Resource	**Campaign Administration**
Target	**Open URL into a new window**
Target Arguments	height=400,width=400
Selected Image	ucf/data/custom/sample/googleOn.gif
Unselected Image	ucf/data/custom/sample/googleOff.gif

Highlighting the Google Web site button

You can highlight the Google Web site button on the Header Bar for specific users. When you highlight a Header Bar button, you display the button in its active (selected) state.

In this procedure, you highlight the button when OEP is launched by the user who has the user ID "sa" (system administrator).

Because this procedure uses the Load event, you must reload the page after completing the procedure to see the changes.

Note: This procedure uses Dynamic Forms Designer. For basic instructions on this window, see Configuring page behavior.

To highlight the Google Web site button:

1. In Dynamic Forms Designer ( ) for the main application frame, add an action statement to the **Load** event on the **Page** object.
2. In the **Description** text box, type: *Highlight the Google Web site button*.
3. Configure the condition as follows:
4. From the **Object** drop-down list, expand **Common** and then select **Get Current User ID**.
5. From the **Operator** drop-down list, select **Equal**; in the **Value** text box, type: *sa*
6. Click in the Summary section to add the condition to the action statement.
7. Configure the action as follows:
   - From the **Object (Action)** drop-down list, expand **Control**, expand **Header_Bar**, and then select **Highlight Item**.
   - From the **Source** drop-down list, select **User Defined**; from the **Value** drop-down box, select **Google**. (Note: Item ID values for this item use the Caption property from Navigation Designer.)
8. Click to confirm changes and then close Dynamic Forms Designer.
9. Click on the UCW toolbar to save all configurations and exit design mode.

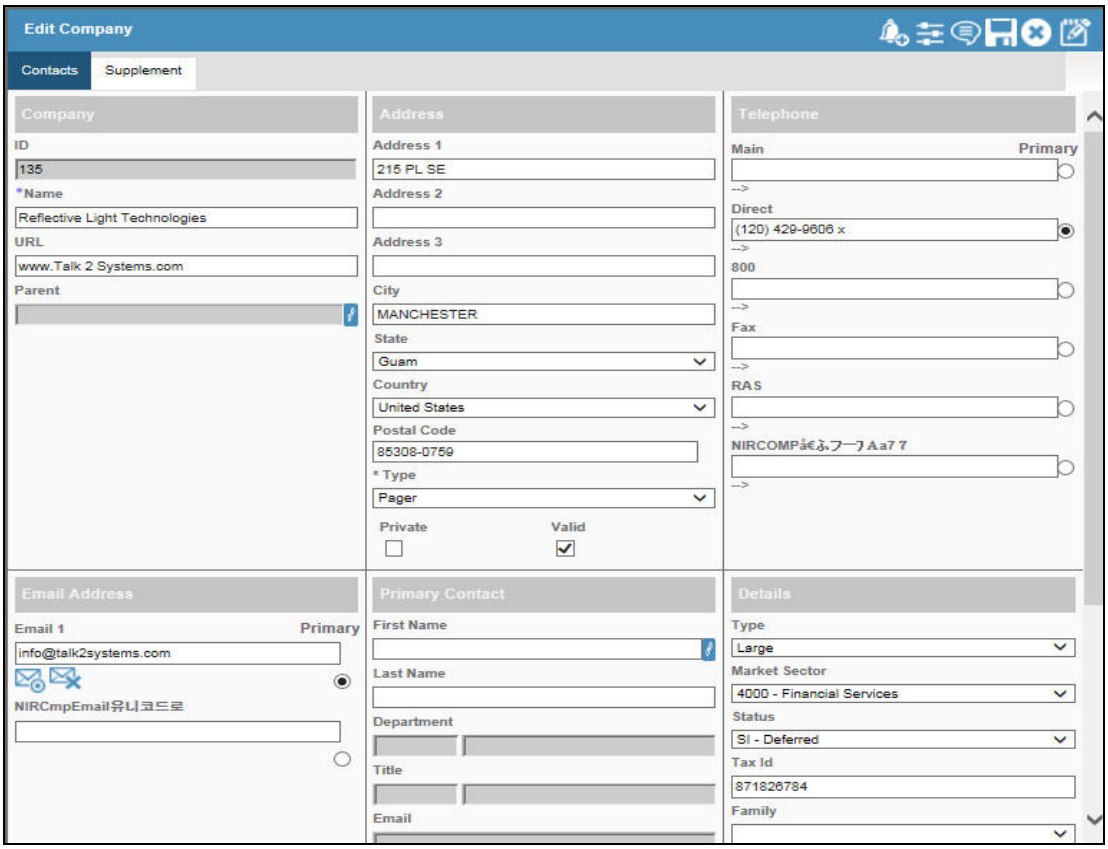
Creating a Tab

Using UCW, you can add tabs to UCW-enabled pages without any programming knowledge.

This sample UI configuration demonstrates adding custom tabs to the page, configuring the tab's layout (panels and sections), adding custom captions, and moving UI controls from one tab to another. This configuration also demonstrates configuring the initially displayed tab for a page.

Additional sample UI configurations in this book follow the scenario presented here to demonstrate adding the following UI elements: a custom toolbar, custom toolbar items (such as buttons), custom UI controls, and UDFs.

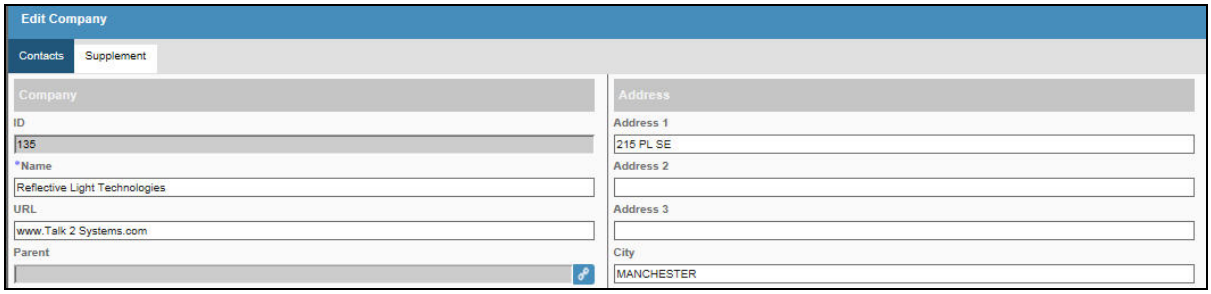
The following graphic illustrates the custom tab on the company edit page as it appears after you complete all procedures in this book. The custom toolbar is placed in the middle of the custom tab, above the "Additional Data" caption. To view the related procedure for a UI element, click the element on the graphic.



Scenario used in this sample UI configuration

During appointments with clients, account managers sometimes change client contact information using the company edit page. The account managers want to remove sensitive data from view if the data does not relate to contact information. They want to move the "Details" sections to a custom tab (with a new title "Detailed Data"), display the company name and source selection in the new tab, and add additional interactive controls to the new tab.

The following graphic illustrates the custom tabs ("Contact" and "Supplemental") and the moved sections under the custom caption "Detailed Data".



Procedures in this topic

Note: These procedures use Navigation Designer, Canvas Designer, and Dynamic Forms Designer. For basic instructions on accessing these windows, see UCW tools.

- Add custom tabs to the page ("Contact" and "Supplemental")
- Configure the layout of the custom Supplemental tab
- Move the Details sections to the Supplemental tab
- Add captions to the Supplemental tab
- (Optional) Configure the initially displayed tab for the company edit page
- Complete the additional procedures that use this scenario

Additional referenced procedures enable you to add a toolbar with buttons (and other items) to the custom tab, add UI controls, and set up UDFs.

Adding custom tabs to the page ("Contact" and "Supplement")

This procedure adds custom tabs titled "Contact" and "Supplement" to the company edit page.

Note: For basic instructions on adding custom tabs, see Configuring tabs.

To add the custom tabs "Contact" and "Supplement" to the company edit page:

- In Navigation Designer ( ) for the company edit page, add tabs to the company edit page with the following properties.

ID	Caption (name)
tabContact	Contact
tabSupp	Supplemental

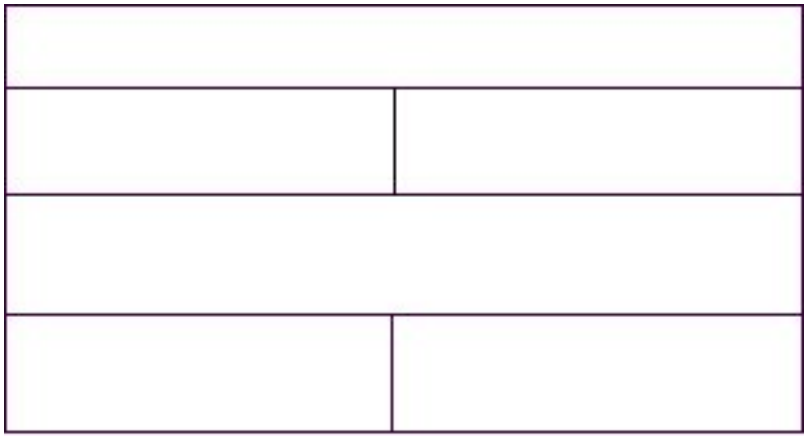
Configuring the layout of the custom Supplement tab

This procedure configures the Supplemental tab created in the previous procedure as a two-panel-by-four-panel layout, with merged panels for a custom toolbar and a custom caption, and a custom TAB key order for the panels.

Note: For basic instructions on configuring layout, see Configuring layout. To learn about Canvas Designer, see Working with Canvas Designer.

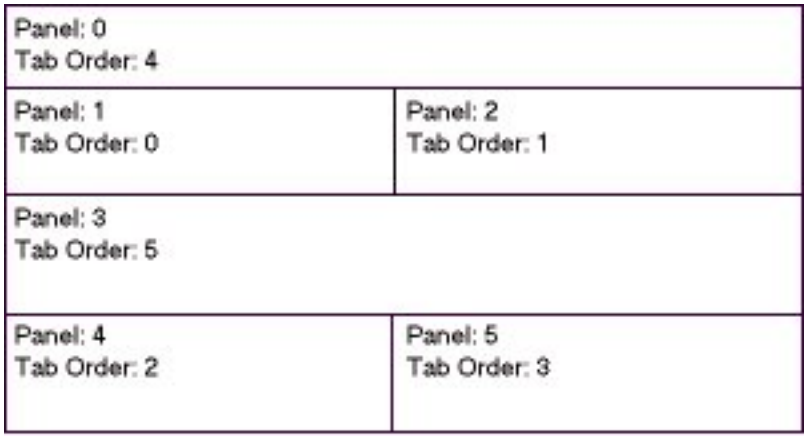
To configure the layout of the Supplemental tab:

1. In the Supplemental tab, right-click the panel's handle ( ) and then select **Convert to Canvas** to display Canvas Designer; confirm the operation if prompted.
2. Add two rows of panels to the default layout and merge the first and third rows to create the layout of panels depicted as follows.



3. Set the TAB key order to go through the split rows in sequence (the controls go in these panels) and then the merged panels in sequence (a caption and a toolbar go in these panels).

The TAB key order is indicated by the "Tab Order" digit, beginning with zero (0), depicted as follows.



4. Confirm changes to the layout and then close Canvas Designer.

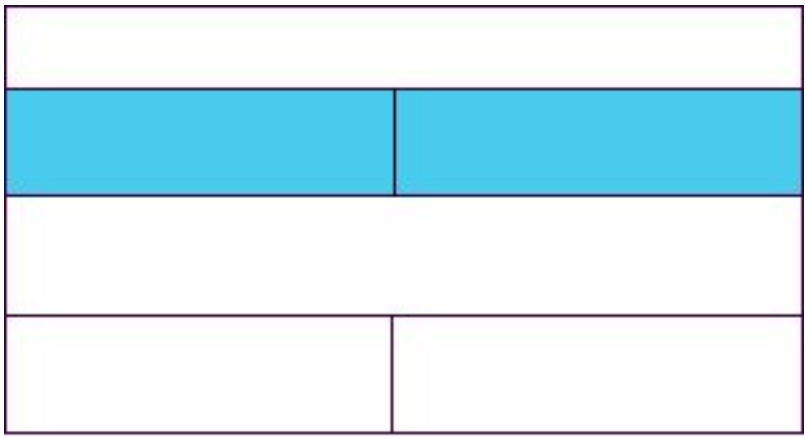
Moving the Details sections to the Supplement tab

This procedure moves the "Details" sections from the Contact tab (original contents of the company edit page) to the Supplemental tab and removes the UI text from the "Details" caption remaining on the Contact tab. These tabs are created in a previous procedure.

Note: To learn about sections, see Working with sections.

To move the Details sections to the Supplement tab:

1. Move each section under the "Details" caption to the Supplemental tab: While dragging each section corner (◤), point to the **Supplement** tab to display the tab, and then continue dragging the section corner to one of the destinations indicated in the following graphic.



All controls contained in the moved sections now appear in the Supplemental tab.

2. Remove the UI text from the existing caption from the Contact tab: Right-click the **Details** caption, select **Properties**, clear the **Caption** text box in the Properties pane, and then click to apply changes.

   The default caption is identified on the page by its Admin ID (because the UI text has been removed). The **Admin ID** property is visible during design mode only; OEP users do not see the **Admin ID** property.

   The next procedure shows how to add the "Detailed Data" caption above the moved sections, along with other custom captions.

Adding captions to the Supplement tab

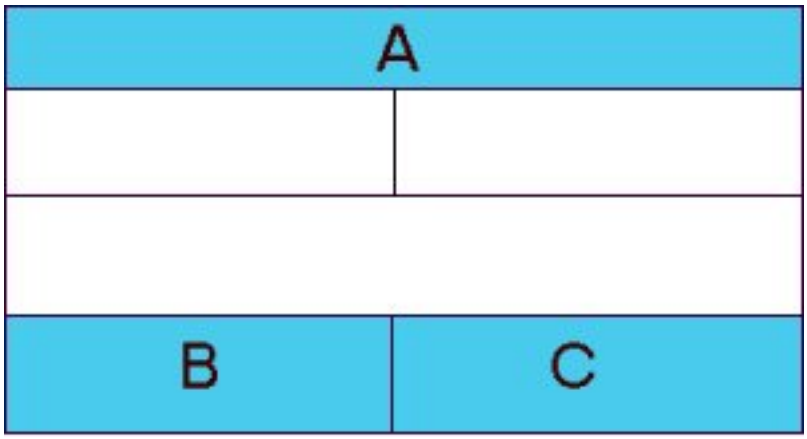
This procedure adds custom captions to panels in the Supplemental tab (created in a previous procedure).

Note: To learn about caption toolbox controls, see Configuring Toolbox controls.

To add captions to the Supplement tab:

- Drag **Caption** from Toolbox Controls to each destination listed in the following table (and indicated in the following graphic) and then type the indicated name in the **Label** text box (located in the Properties pane).

Destination	Admin ID	Label
"A" - Panel above the sections moved from the other tab	*Caption_DetailedData*	*Detailed Data*
"B" - Bottom row, left panel	*Caption_AdditionalData*	*Additional Data*
"C" - Bottom row, right panel	*Caption_UDFs*	*UDFs*



Configuring the initially displayed tab for the company edit page (optional)

This procedure configures the Supplemental tab (created in a previous procedure) as the initially displayed tab for users who log on to OEP with the selected profile. Completion of this procedure is not required for the additional procedures.

Because this procedure uses the Load event, you must reload the page after completing the procedure to see the changes.

Note: This procedure uses Dynamic Forms Designer. For information on this window, see Configuring page behavior.

To configure the initially displayed tab for the company edit page:

1. In Dynamic Forms Designer ( ) for the company edit page, add an action statement to the **Load** event on the **Page** object; type the description *Display Supplemental tab by default*.
2. In the **Object (Action)** drop-down list located in the Actionsection, expand **Control**, expand **Top_Tab**, and then select **Select Item**.
3. From the **Item ID Source** drop-down list, select **User Defined**; from the **Value** drop-down list, select **Supplemental**.
4. Click to confirm changes and then close Dynamic Forms Designer.
5. Click on the UCW toolbar to save all configurations and exit design mode.

Additional sample UI configurations using the custom Supplemental tab

The following topics in this book contain procedures that use the custom Supplemental tab created within this topic for the company edit page.

- Add a custom toolbar to the Supplemental tab
- Add buttons and other items to the custom toolbar
- Add UI controls to the Supplement tab
- Add UDFs to the Supplement tab

Adding a Toolbar

Using UCW, you can add a toolbar to UCW-enabled pages without any programming knowledge.

This sample UI configuration demonstrates adding a custom toolbar to the page.

Scenario used in this sample UI configuration

The account managers tell you they want quick access to Google.com and the ability to hide or show the primary tab from the custom (secondary) tab. You decide to add these functions to a custom toolbar at the top of the Supplemental tab.

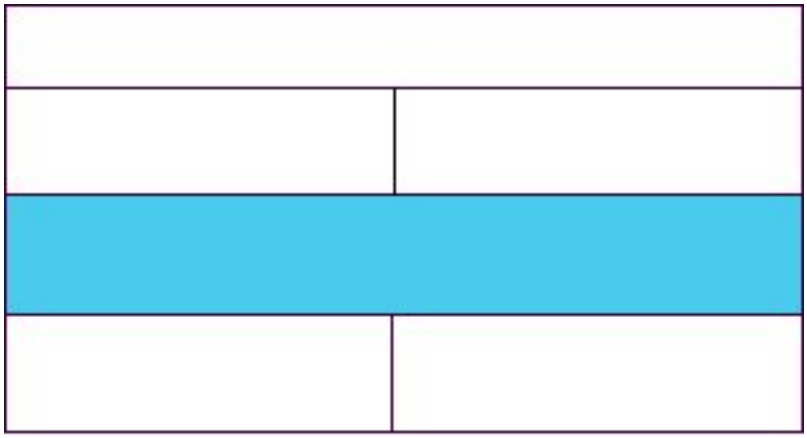
This procedure adds a custom toolbar. The functions (toolbar items) are added in the next sample UI configuration.

The following graphic illustrates the custom toolbar with its title "Supplemental". In this graphic, the custom toolbar buttons and other items have not yet been added.

Note: For basic information on adding toolbars, see Configuring toolbars

To add a custom toolbar to the Supplemental tab:

1. Follow the procedure for creating the custom Contact and Supplemental tabs on the company edit page.
2. In Navigation Designer ( ) for the company edit page, add a toolbar with the following **ID** property: toolbarSupp
3. Place a toolbar toolbox control in the third row (second merged panel) on the Supplemental tab:



4. Enter the following property configuration (typed entries are indicated by italics; selected entries are indicated by boldface).

   The Association ID property references the toolbar configuration created in Navigation Designer. The Admin ID property is used in the Adding toolbar buttons procedure.

Property	Value
Admin ID	Toolbar_Supplemental
Tooltip Text	Toolbar for the Supplemental tab
Association ID	toolbarSupp
Title	Supplemental

Click here to go to the next sample UI configuration, which adds toolbar buttons and other items to the custom toolbar.

Adding Toolbar Buttons and Other Items

Using UCW, you can add toolbar buttons and other toolbar items to toolbars on UCW-enabled pages without any programming knowledge.

This sample UI configuration demonstrates adding toolbar items of the following types to a custom toolbar: Button, MultiState, and Action Menu. This topic also demonstrates adding functionality to toolbar items.

The Expand/Contract toolbar item type is excluded from this example.

Scenario used in this sample UI configuration

The account managers tell you they want quick access to Google.com and the ability to hide or show the primary tab from the custom (secondary) tab. You decide to add these functions to a custom toolbar at the top of the Supplemental tab. (The custom toolbar was created and added to the page in the previous sample UI configuration.)

The following graphic illustrates toolbar items on the custom "Supplemental" toolbar:

Procedures in this sample UI configuration

- Add toolbar items to the Supplemental toolbar
- Add functionality to the Button toolbar item to launch Google.com
- Add functionality to the MultiState toolbar item to hide and show the Contact tab

Note: For basic information on adding toolbar items, see Configuring toolbars.

Adding toolbar items to the Supplemental toolbar

This procedure adds Button, MultiState, and Action Menu toolbar items to the Supplemental toolbar.

Note: The Button and MultiState toolbar items fulfill the needs defined in the scenario for this sample UI configuration. The procedure includes the Action Menu toolbar item for demonstration purposes.

To add toolbar items to the Supplemental toolbar:

1. Follow the procedure for adding the Supplemental toolbar. (This procedure assumes completion of the steps in the Creating a tab topic.)
2. Copy the sample images from the ../TechnicalGuide/Examples/Images/Toolbar directory on the OEP product CD to the following directory on your OEP Web server: *YourOEPwebsite*/ucf/data/custom/sample

Note: Onyx recommends creating a company-specific subdirectory under YourOEPwebsite/ucf/data/custom/ for your own images.

3. In Navigation Designer ( ) for the company edit page, add the following toolbar items and values to the Supplemental toolbar (identified as "toolbarSupp"), using the indicated configuration entries (typed entries are indicated by italics; selected entries are indicated by boldface). If properties are not listed in the table, leave the default entries.

Type	ID (toolbar item)	ID (toolbar item value)	Property	Entry
Button (one value)	*toolbarItemBtn*		**Type**	*ucf/data/custom/sample/iconGoogle0.gif*
		btnGoogle	**Tooltip**	*Hide Contact tab*
			Image	*ucf/data/custom/sample/iconnumber200.gif*
MultiState (two values)	*toolbarItemHideShow*		**Type**	*Show Contact tab*
		iconHide	**Value**	*ucf/data/custom/sample/iconnumber250.gif*
			Tooltip	*Choose the number of widgets*
			Image	**Image Only**
		iconShow	**Value**	**Image and Text**
			Tooltip	*5 widgets*
			Image	*5*
Action Menu (drop-down list; three values)	*toolbarItemWidgets*		**Type**	*ucf/data/custom/sample/iconnumber050.gif*
			Tooltip	*10 widgets*
			Display Mode	*10*
			List Mode	*ucf/data/custom/sample/iconnumber100.gif*
		option05	**Value**	*15 widgets*
			Tooltip	*15*
			Caption	*ucf/data/custom/sample/iconnumber150.gif*
			Image	*ucf/data/custom/sample/iconGoogle0.gif*

Type	ID (toolbar item)	ID (toolbar item value)	Property	Entry
		option10	**Value**	*Hide Contact tab*
			Tooltip	*ucf/data/custom/sample/iconnumber200.gif*
			Caption	*Show Contact tab*
			Image	*ucf/data/custom/sample/iconnumber250.gif*
		option15	**Value**	*Choose the number of widgets*
			Tooltip	**Image Only**
			Caption	**Image and Text**
			Image	*5 widgets*

Adding functionality to the Button toolbar item (launching facebook.com)

This procedure adds functionality to the Button toolbar item on the Supplemental toolbar. The functionality consists of launching Google.com in a separate browser window with a specified height and width of 400 pixels.

To add functionality to the Button toolbar item (launch facebook.com):

1. In Dynamic Forms Designer ( ) for the company edit page, add an action statement to the Item Click event for the toolbar: Expand **Control** in the Events pane, expand **Toolbar_Supplemental** (the toolbar's **Admin ID** property), and then select the **Item Click (Item ID) (Item Value)** event.
2. In the **Description** text box, type: Launch Google.com with toolbarItemBtn
3. Add a condition to specify the Button toolbar item (toolbarItemBtn):
   a. In the **Object** drop-down list located in the Conditions section, expand **Event Arguments** and then select **Item ID**.
   b. From the **Operator** drop-down list, select **Equal**; in the **Value** text box, type: toolbarItemBtn
   c. Click in the Summary section to add the condition to the action statement.

   The toolbar item value does not need to be specified because the Button toolbar item contains only one value.
4. Configure the action to launch Facebook.com as follows:
   a. From the **Object (Action)** drop-down list, expand **Common**, expand **UI**, and then select **Open New Window**.
   b. From each **Source** drop-down list, select **User Defined**; in the **Value** text boxes, type the following entries:

Text box	Entry
URL	*http://www.google.com*
Target	*_blank*
Features	*height=400,width=400*

5. Click to confirm changes.

Adding functionality to the MultiState toolbar item (hiding and showing tabs)

This procedure adds functionality to each toolbar item value for the MultiState toolbar item on the Supplemental toolbar. The functionality consists of hiding and showing the Contact tab on the company edit page.

To add functionality to the MultiState toolbar item (hide and show a tab):

1. In Dynamic Forms Designer ( ) for the company edit page, add an action statement to the Item Click event for the toolbar: Expand **Control** in the Events pane, expand **Toolbar_Supplemental** (the toolbar's **Admin ID** property), and then select the **Item Click (Item ID) (Item Value)** event.
2. Add the following conditions to specify the MultiState toolbar item (toolbarItemHideShow) and the iconHide toolbar item value (1):
   a. In the **Object** drop-down list located in the Conditions section, expand **Event Arguments** and then select **Item ID**.
   b. From the **Operator** drop-down list, select **Equal**; in the **Value** text box, type: *toolbarItemHideShow*
   c. Click in the Summary section to add the condition to the action statement.
   d. In the **Object** drop-down list located in the Conditions section, expand **Event Arguments** and then select **Item Value**.
   e. From the **Operator** drop-down list, select **Equal**; in the **Value** text box, type: *1*
   f. Click the top-most in the Summary section (above the "OR" operator) to add the condition to the action statement.
   g. Both conditions must be satisfied for the action to occur; the conditions have an implicit "AND" operator between them.
3. Configure the action to hide the Contact tab as follows:
   a. From the **Object (Action)** drop-down list, expand **Control**, expand **Top_Tab**, and then select **Hide Item**.
   b. From the **Source** drop-down list, select **User Defined**; from the **Value** drop-down list, select **Contact**.
4. Click to confirm changes.

a. Add another action statement to the Item Clickevent.

b. Add the following conditions to specify the MultiState toolbar item (toolbarItemHideShow) and the iconShow toolbar item value (2):

c. In the **Object** drop-down list located in the Conditions section, expand **Event Arguments** and then select **Item ID**.

d. From the **Operator** drop-down list, select **Equal**; in the **Value** text box, type: *toolbarItemHideShow*

e. Click  in the Summary section to add the condition to the action statement.

f. In the **Object** drop-down list located in the Conditions section, expand **Event Arguments** and then select **Item Value**.

g. From the **Operator** drop-down list, select **Equal**; in the **Value** text box, type: *2*

h. Click the top-most  in the Summary section (above the "OR" operator) to add the condition to the action statement.

Both conditions must be satisfied for the action to occur; the conditions have an implicit "AND" operator between them.

5. Configure the action to show the Contact tab as follows:

a. From the **Object (Action)** drop-down list, expand **Control**, expand **Top_Tab**, and then select **Show Item**.

b. From the **Source** drop-down list, select **User Defined**; from the **Value** drop-down list, select **Contact**.

6. Click  to confirm changes.

7. To confirm toolbar item functionality, save all configurations and exit design mode (click  on the UCW toolbar) and then redisplay the company edit page.

Adding UI Controls

Using UCW, you can add custom UI controls to UCW-enabled pages. You can bind UI controls to any property in the Onyx Enterprise Dictionary (OED) for the current page object.

This sample UI configuration demonstrates adding user-interactive UI controls (Dropdown, Textbox, and Date) and display-only UI controls (Text and Text Reference) to the page and binding the controls to default and custom properties. This topic also describes the steps required to define custom properties and lookup values.

Scenario used in this sample UI configuration

The account managers ask you to create new UI controls for the Supplemental tab you added to the company edit page (see Creating a tab). The account managers want to display the company name and selected source on the custom tab. They also want to add controls for their employees to select an ownership type (private or public), enter the number of employees, and enter or select the date on which the company was founded.

The following graphic illustrates the custom UI controls under the custom caption "Additional Data".

Procedures in this sample UI configuration

- Add display-only UI controls (bound to default properties)
- Define custom properties and lookup values
- Add user-interactive UI controls (bound to custom properties)

> **Note:** For basic information on UI controls and properties, see Configuring UI controls and Control properties reference.

Adding display-only UI controls to the page

This procedure adds the display-only UI controls required to display the company name and source in the Supplemental tab, using the toolbox control types **Text** and **Text Reference**. Both controls are data-bound to default properties in the OED, and the **Text Reference** control also references a default field for lookup values.

> **Note:** Default properties are used here for demonstration purposes. You can bind custom UI controls to any property in the OED, whether default or custom.

To add display-only UI controls to the Supplemental tab, bound to default properties:

1. Create a custom UI control that is bound to the property for the company name: Add a **Text** toolbox control to the section under the Additional Data caption in the Supplemental tab on the company edit page, using the following configuration entries (typed entries are indicated by italics; selected entries are indicated by boldface).

Property	Entry
Admin ID	*toolboxName*
Tooltip Text	*Lists the company name entered in the Contact tab*
Label	*Name*
Object	**Company**

Property	Entry
Property	**companyName**
Mode	**Read-only**

Note: Read-only is the typical Mode setting for display-only controls. For more information, see Control properties reference.

2. Move the intrinsic control labeled "Source" from the Detailed Data section of the Supplemental tab to an empty panel in the Contact tab.

   The next custom UI control displays the data for this control.

3. Create a custom UI control that is bound to the property for the source and that refers to the corresponding field for its lookup values: Add a **Text Reference** toolbox control below the existing Name toolbox control (in the section under the Additional Data caption), using the following configuration entries (typed entries are indicated by italics; selected entries are indicated by boldface).

Property	Entry
Admin ID	*toolboxSource*
Tooltip Text	*Lists the source entered in the Contact tab*
Label	*Source*
Field Name	*company.source*
Object	**Company**
Property	**sourceId**
Mode	**Read-only**

Note: Read-only is the typical Mode setting for display-only controls. For more information, see Control properties reference.

4. Click  on the UCW toolbar to save all configurations and exit design mode.
5. Go to the next procedure to define custom properties and lookup values.

Defining custom properties and lookup values

This procedure defines the custom properties required for the requested controls: ownership type (private or public), number of employees, and date on which the company was founded. This procedure also defines the "Private" and "Public" lookup values for the ownership type.

A database administration tool is required to add columns to the database table.

Note: For information on properties, see OEAS Technical Reference; for information on field and lookup values, see *OEAS Technical Reference*.

To define custom properties and lookup values:

1. Add columns to the database table that represents the company object, using a database administration tool and the following configuration entries.

Column name (database)	Data type	Allow Nulls?
Ownership	*text*	*Yes*
number_empl	*int*	*Yes*
date_founded	*datetime*	*Yes*

2. Add corresponding columns to the company object physical model in the OED (Onyx Enterprise Dictionary), using OES Object Designer and the following configuration entries (typed entries are indicated by italics; selected entries are indicated by boldface). Note: Other than case, the OED column name must be identical to the database column name.

Name (OED column)	Description	Column Type	Length
OWNERSHIP	*Type of ownership for the company.*	**TEXT**	*255*
NUMBER_EMPL	*The number of employees working at the company.*	**INT**	*5*
DATE_FOUNDED	*The date on which the company was founded.*	**DATETIME**	(leave blank)
Name (OED column)	Description	Column Type	Length

3. Add corresponding properties to the company object logical business object (LBO) in the OED, using OES Object Designer and the following configuration entries (typed entries are indicated by italics; selected entries are indicated by boldface).

Name (property)	Description	Data Type
ownership	*Type of ownership for the company.*	**string**
numberEmpl	*The number of employees working at the company.*	**int**
dateFounded	*The date on which the company was founded.*	**dateTime**

4. Add object physical maps to the company object logical business object (LBO) to connect the new logical properties to the new physical columns in the company table, using OES Object Designer and the following configuration entries.

Property	Table	Column	Maps (Persistence Map, Retrieval Map, Filter Map)	Type (for all enabled maps)
ownership	**COMPANY**	**OWNERSHIP**	**Enable**	**direct**
numberEmpl	**COMPANY**	**NUMBER_ EMPL**	**Enable**	**direct**
dateFounded	**COMPANY**	**DATE_ FOUNDED**	**Enable**	**direct**

5. Publish the changes to the OED:
   a. Click  to save changes to the OED.
   b. To validate the changes first, click  on the upper right of the OES window.
   c. Click  on the upper right of the OES window to publish the changes; when prompted, confirm the operation.
6. Shut down the Onyx EAS COM+ application (OnyxEASApp), using an administrative tool such as "Component Services." (This action causes the OED to be reloaded the next time a client calls the OEAS.)
7. Add a field and lookup values to the company object for the ownership property, using OES Reference Table Administration.

Note: For information on field and lookup values, see the OEAS Technical Reference.

   a. Select Fields on the upper left of the OES window, select the **company** object, and then click  to add a field with the following configuration entries (typed entries are indicated by italics; selected entries are indicated by boldface).

Note: The Sequence # entry is ignored on UCW-configured pages; however, an entry is required for new fields.

Field Name	Caption	Sequence #	Field Type
company.ownership	*Ownership*	1	**Combo Box**

   b. Click  to save the field.
   c. Select **Lookup Value by Field** on the upper left of the OES window, expand the **company** object, select the **company.ownership** field, and then click  to add a lookup value

d. In the **Lookup Value** textbox on the upper right of the window, type **Private**; click **Add Lookup Value** to add another lookup value.

e. In the **Lookup Value** textbox on the upper right of the window, type **Public**; click  to save changes.

8. Stop and restart the Web server for OEP. Note: You can stop and restart Web servers using administrative tools such as the Internet Information Services (IIS) Manager.
9. Go to the next procedure to add interactive UI controls to the page for these custom properties and lookup values.

Adding user-interactive UI controls to the page

This procedure adds the user-interactive UI controls required to give users the ownership type options (private and public) and properly formatted text boxes for entering the number of employees and the date on which the company was founded, using the toolbox control types **Dropdown**, **Textbox**, and **Date**, respectively, and placing the controls in the Supplemental tab on the company edit page. This procedure also data-binds the controls to custom properties in the OED and, for the **Dropdown** control, references a custom field for the "Public" and "Private" lookup values used for the ownership type options.

 **Note:** Custom properties are used here for demonstration purposes. You can bind custom UI controls to any property in the OED (for the page-specific object), whether default or custom.

To add user-interactive UI controls, bound to custom properties:

1. Define the custom properties and lookup values.
2. Add a **Dropdown** toolbox control below the existing Source toolbox control on the page (under the Additional Data caption), using the following configuration entries (typed entries are indicated by italics; selected entries are indicated by boldface).

Property	Entry
Admin ID	*toolboxOwnership*
Tooltip Text	*Select the type of ownership for this company*
Label	*Ownership*
Field Name	*company.ownership*
Object	**Company**
Property	**Ownership**
Mode	**Read/Write**

3. Add the following toolbox controls below the existing Ownership toolbox control, using the following configuration entries (typed entries are indicated by italics; selected entries are indicated by boldface).

Property	Entry (by type of toolbox control)	
Date	Textbox	Date
Admin ID	*toolboxNumberEmp*	*toolboxDateFounded*
Tooltip Text	*Enter the number of employees working at this company*	*Enter the date on which this company was founded*
Label	*Number of employees*	*Date founded*
Object	**Company**	**company**
Property	**numberEmpl**	**dateFounded**
Mode	**Read/Write**	**Read/Write**

4. Set the five-character maximum length for the Textbox toolbox control ("Number of employees"). For instructions, see Validating maximum length (Textbox toolbox controls).
5. Click  on the UCW toolbar to save all configurations and exit design mode.

Setting up UDFs

Using UCW, you can add previously configured UDFs, or User Defined Fields, to UCW-enabled pages.

Note: For troubleshooting information on binding toolbox controls to UDFs, see the related section in Troubleshooting configuration.

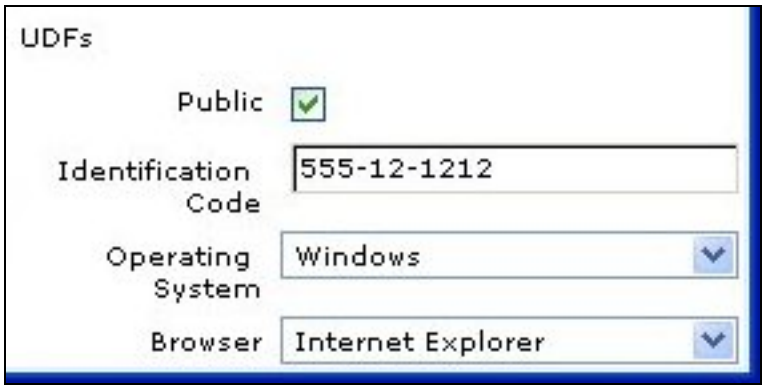
Use OES Reference Table Administration to create and configure UDFs. UDFs are useful for creating check box controls, masked controls, and hierarchically related controls.

This sample UI configuration demonstrates configuring UDFs in OES and then using UCW to place UDFs on a tab.

Scenario used in this sample UI configuration

The account managers ask you to create additional UI controls for the Supplemental tab you added to the company edit page (see Creating a tab). These controls include a check box to indicate whether or not the information is public, a text box for entry of an identification code that automatically adds intervening dashes, and a hierarchical set of drop-down lists for selection of operating systems and compatible browsers. Because toolbox controls do not include check boxes, masked text boxes, or hierarchical drop-down lists, you decide to use UDFs to satisfy this request.

The following graphic illustrates the UDFs under the custom caption "UDFs".



Procedures in this sample UI configuration

- Configure UDFs in OES
- Add UDFs to the page
- Configure the toolbar button to clear all UDFs (optional)

Configuring UDFs in OES (check box, masked text box, and hierarchical sets of drop-down lists)

This procedure configures the UDFs in OES, using the formats required for a check box, a masked text box, and a hierarchical set of drop-down lists.

Note: The following settings in OES Reference Table Administration are treated differently on UCW-configured pages. Sequence # (TAB key order) is ignored and captions are overridden by UCW-defined captions (for the selected profile). For basic information on UDFs, see OEAS Technical Reference and OEAS Technical Reference.

To configure the check box UDF in OES:

1. In OES Reference Table Administration, select the **Fields** mode option, and then expand the **company** node to display its UDFs.
2. Select **company.user1** and configure the check box UDF using the following configuration entries (typed entries are indicated by italics; selected entries are indicated by boldface).

Property	Entry
Caption	*Public*
Active	Yes (checked)
Field Type	**Check Box**

3. Go to the next procedure to configure a masked UDF.

To configure the masked UDF in OES:

1. Select **company.user2** and configure the masked UDF using the following configuration entries (typed entries are indicated by italics; selected entries are indicated by boldface). Properties not listed can retain their default values.

Property	Entry
Caption	*Identification Code*
Active	Yes (checked)
Field Type	**Text Box**
Mask	*999-99-9999*

2. Go to the next procedure to configure a set of hierarchically related UDFs.

To configure the set of hierarchically related UDFs in OES:

1. Select **company.user3** and configure the "parent" UDF using the following configuration entries (typed entries are indicated by italics; selected entries are indicated by boldface).

Property	Entry
Caption	*Operating System*
Active	Yes (checked)
Field Type	**Combo Box**

2. Select **company.user4** and configure the "child" UDF using the following configuration entries (typed entries are indicated by italics; selected entries are indicated by boldface).

Property	Entry
Caption	*Browser*
Active	Yes (checked)
Field Type	**Combo Box**
Parent Field	**company.user3**

3. Click  to save the new fields.
4. To create lookup values for the "Operating System" UDF:
   a. Select **Lookup Value by Field** on the upper left of the OES window, expand the **company** object, select the **company.user3** field, and then click  to add a lookup value.

b. In the **Lookup Value** textbox on the upper right of the window, type Windows; click **Add Lookup Value** (in the right pane) to add another lookup value.

c. In the **Lookup Value** textbox on the upper right of the window, type Mac; click to save changes.

4. To create "Browser" UDF lookup values corresponding to the "Windows" option in the "Operating System" UDF:

a. Expand **company.user3**, expand **Windows**, select **company.user4**, and then click to add a lookup value.

b. In the **Lookup Value** textbox on the upper right of the window, type Internet Explorer; click **Add Lookup Value** (in the right pane) to add another lookup value.

c. In the **Lookup Value** textbox on the upper right of the window, type Lynx; click to save changes.

5. To create "Browser" UDF lookup values corresponding to the "Mac" option in the "Operating System" UDF:

a. Expand **company.user3**, expand **Mac**, select **company.user4**, and then click to add a lookup value.

b. In the **Lookup Value** textbox on the upper right of the window, type *Mozilla*; click **Add Lookup Value** (in the right pane) to add another lookup value.

c. In the **Lookup Value** textbox on the upper right of the window, type Safari; click to save changes.

6. Stop and restart the Web server for OEP.

Note: You can stop and restart Web servers using administrative tools such as the Internet Information Services (IIS) Manager.

Place UDFs on the Supplemental tab

You can place UDFs on the Supplemental tab of the company edit page (created in the procedure Creating a tab). Active UDFs are listed in the Intrinsic Controls pane. Note: To activate a UDF, select the Active check box in OES Reference Table Administration.

Note: For information on UDF control properties, see Control properties reference.

To add the UDFs to the Supplemental tab:

1. Drag the pre-configured UDFs (Public, Identification Code, Operating System, and Browser) from the Intrinsic Controls pane to the section under the UDFs caption in the Supplemental tab on the company edit page.

UDFs are identified in the Intrinsic Controls pane by the labels pre-configured in OES.

2. Configure the UDF's Admin ID properties and tooltips by changing properties as indicated in the following table (typed entries are indicated by italics; selected entries are indicated by boldface).

Label (pre-configured in OES)	Admin ID	Tooltip Text
Public	UDF_Public	*Information not sensitive when checked*
Identification Code	**UDF_ID**	*Identifier*
Operating System	**UDF_OS**	*Most prevalent operating system used by employees*
Browser	**UDF_Browser**	*Most prevalent browser used by employees*

3. Either go to the next (optional) procedure to configure a toolbar button for clearing UDFs or click  on the UCW toolbar to save all configurations and exit design mode.

Configure the toolbar button to clear all UDFs (optional)

You can configure a toolbar button to clear the values of the specified controls.

This procedure configures the existing toolbar button (previously configured to launch Google.com) to clear the values of the UDFs on the Supplemental tab. This procedure also demonstrates the usefulness of cloning statements.

To configure the toolbar button to clear all UDFs:

1. Click  on the UCW toolbar for the company edit page to display Dynamic Forms Designer.
2. Deactivate or delete the action statement that configures the toolbar button to launch Google.com:
   a. Expand **Control** in the Events pane, expand **Toolbar_Supplemental** (the toolbar's **Admin ID** property), expand the **Item Click (Item ID) (Item Value)** event, and then select the action statement identified by the description "Launch Google.com with toolbarItemBtn" (as entered in a previous UI configuration).
   b. Choose an action:
      - To deactivate the statement, clear the **Active** check box and then click  to confirm changes.
      - To delete the statement, click  .
3. Add an action statement to the **Item Click (Item ID) (Item Value)** event for **Toolbar_ Supplemental**.
4. In the **Description** text box, type: *Clear UDF_Public*
5. Add a condition to specify the Button toolbar item (*toolbarItemBtn*):

a. In the **Object** drop-down list located in the Conditions section, expand **Event Arguments** and then select **Item ID**.

b. From the **Operator** drop-down list, select **Equal**; in the **Value** text box, type: *toolbarItemBtn*

c. Click in the Summary section to add the condition to the action statement.

6. Configure the action to clear the UDF_Public control as follows:

a. From the **Object (Action)** drop-down list, expand **Control**, expand **UDF_Public**, and then select **Set Value**.

b. From each **Source** drop-down list, select **User Defined**; from the **Value** drop-down list, select **Not Checked**.

7. Click to confirm changes.
8. Clone the statement three times.

The cloned statements appear below the original statement under the **Item Click (Item ID) (Item Value)** event for **Toolbar_Supplemental**.

9. For each cloned statement, change the description and action to be specific to a different UDF control as indicated in the following table, and then click to confirm changes. (Typed entries are indicated by italics; selected entries are indicated by boldface).

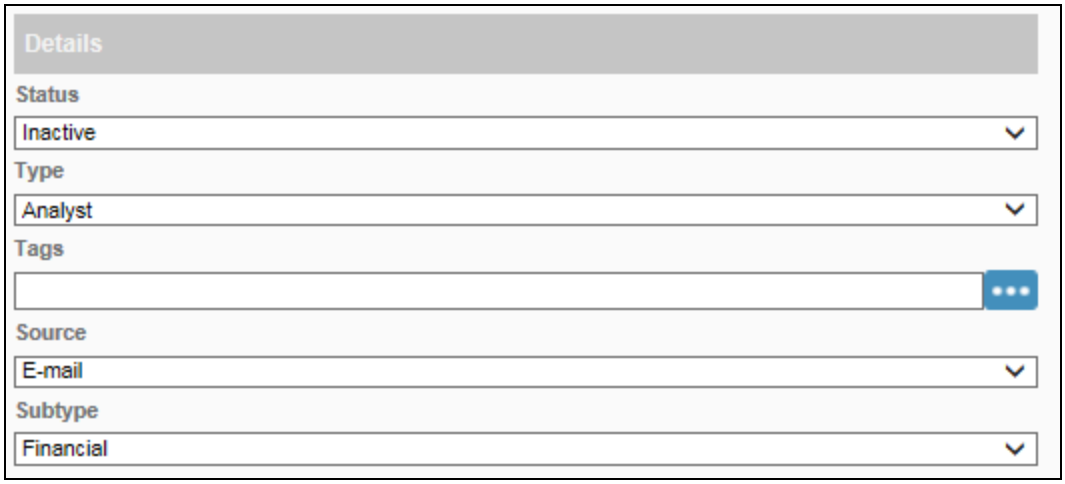
Cloned statement (position below original statement)	Description	Object (Action)	Source	Value
Second	*Clear UDF_ID*	**UDF_ID (Set Value)**	**User Defined**	(empty)
Third	*Clear UDF_OS*	**UDF_OS (Set Value)**	**User Defined**	(empty)
Fourth	*Clear UDF_ Browser*	**UDF_Browser (Set Value)**	**User Defined**	(empty)

Use Navigation Designer ( ) to change the toolbar button's properties, such as its tooltip or image.

10. Close Dynamic Forms Designer and then click on the UCW toolbar to save all configurations and exit design mode.

Congratulations! You have completed all sample UI configurations in the Creating a tab book.

The following graphic illustrates the custom tab on the company edit page as it appears after you complete all procedures in this book. To view the related procedure for a UI element, click the element on the graphic.



Setting Default Values

Using UCW, you can set default values for controls on UCW-enabled pages without any programming knowledge.

This sample UI configuration demonstrates setting a default value for a control when a new record is displayed or when another control's value is changed.

Scenario used in this sample UI configuration

The marketing department wants all new company records and records changed to large company subtypes to automatically indicate an internal source (Internal value for the Source control). This automatic selection reduces the number of required keystrokes.

Procedures in this sample UI configuration

Note: These procedures use Dynamic Forms Designer. For basic instructions on using Dynamic Forms Designer, see Adding and modifying action statements.

Follow these procedures to set the default value for Source in new or large company records.

- Set Source to Internal for new records
- Set Source to Internal for large company records
- Set Source to empty for other records
- Save all configurations and exit design mode

Setting Source to Internal for new records

You can dynamically set the default value for the Source control to Internal when the OEP user displays an "empty" company edit page for addition of a new record.

The condition in this procedure checks for an "empty" page by determining the value of the primary ID; if the ID is empty, the page is also empty (containing no record).

Because this procedure uses the Load event, you must reload the page after completing the procedure to see the changes.

To set Source to Internal if the record is new:

1. In Dynamic Forms Designer ( ) for the company edit page, add an action statement to the **Load** event on the **Page** object.
2. In the **Description** text box, type *Internal source for new records*
3. Configure the condition as follows:
   a. From the **Object** drop-down list, expand **Context_Data**, expand **Primary ID**, and then select **Get Value**.
   b. From the **Operator** drop-down list, select **Is Empty**.
   c. Click in the Summary section to add the condition to the action statement.
4. Configure the action as follows:
   a. From the **Object (Action)** drop-down list, expand **Control**, expand **Source**, and then select **Set Value**.
   b. From the **Source** drop-down list, select **User Defined**; from the **Value** drop-down list, select **Internal**.
5. Click to confirm changes.

Setting Source to Internal for large customer records

You can dynamically set the default value for the Source control to Internal when the OEP user changes the Subtype control to Large Account.

This procedure configures an action statement on the Change event for the Subtype control.

To set Source to Internal when the record subtype is changed to Large Account:

1. In Dynamic Forms Designer ( ) for the company edit page, add an action statement to the **Change** event for the **Subtype** control object (under the **Control** node).
2. In the **Description** text box, type Internal source for Large Account records
3. Configure the condition as follows:

a. From the **Object** drop-down list, expand **Control**, expand **Subtype**, and then select **Get Value**.

b. From the **Operator** drop-down list, select **Equal**; from the **Value** drop-down list, click  to display the Hierarchical Value Selector window.

c. From the **Type** drop-down list under Parent Control(s), select **Customer**; from the Subtype drop-down list under Selected Child Control, select **Large Account**.

d. Click  to confirm changes and close the Hierarchical Value Selector window.

e. Click  in the Summary section to add the condition to the action statement.

4. Configure the action as follows:

a. From the **Object (Action)** drop-down list, expand **Control**, expand **Source**, and then select **Set Value**.

b. From the **Source** drop-down list, select **User Defined**; from the **Value** drop-down list, select **Internal**.

5. Click  to confirm changes.

Setting Source to empty for other records

You can dynamically set the default value for the Source control to empty (no value) when the OEP user changes the Subtype control to a value other than Large Account.

This procedure configures an action statement on the Change event for the Subtype control.

To set Source to empty (no value) when the record subtype is changed to a value other than Large Account:

1. In Dynamic Forms Designer () for the company edit page, add an action statement to the **Change** event for the **Subtype** control object (under the **Control** node).
2. In the **Description** text box, type *Empty source for other records*
3. Configure the condition as follows:

a. From the **Object** drop-down list, expand **Control**, expand **Subtype**, and then select **Get Value**.

b. From the **Operator** drop-down list, select **Not Equal**; from the **Value** drop-down list, click  to display the Hierarchical Value Selector window.

c. From the **Type** drop-down list under Parent Control(s), select **Customer**; from the **Subtype** drop-down list under Selected Child Control, select **Large Account**.

d. Click  to confirm changes and close the Hierarchical Value Selector window.

e. Click  in the Summary section to add the condition to the action statement.

4. Configure the action as follows:

   a. From the **Object (Action)** drop-down list, expand **Control**, expand **Source**, and then select **Set Value**.

   b. From the **Source** drop-down list, select **User Defined**; leave the **Value** entry empty.

5. Click  to confirm changes.

Saving all configurations and exiting design mode

When you complete all procedures in this topic, you must save the configurations.

To save all configurations and exit design mode:

- Close Dynamic Forms Designer and then click  on the UCW toolbar to save all configurations and exit design mode.

Setting Controls as Required

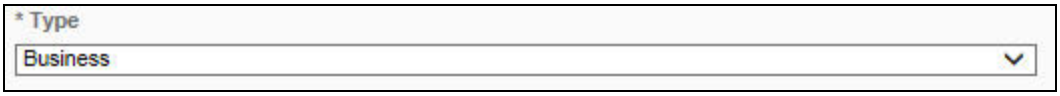
Using UCW, you can set controls on UCW-enabled pages to required without any programming knowledge.

This sample UI configuration demonstrates setting a control as required by configuring action statements that depend on the user-selected values for another set of controls.

Scenario used in this sample UI configuration

The manager of the remote sales team asks you to make the Type control a required entry when her staff adds or edits "active lead" records (on the individual edit page). This requirement ensures that the salespeople remember to select a hot, warm, or cold status for active leads.

The following graphic illustrates the Type control with an asterisk displayed to indicate that a selection is required.



Procedures in this sample UI configuration

Note: These procedures use Dynamic Forms Designer. For basic instructions on using Dynamic Forms Designer, see Adding and modifying action statements.

Follow these procedures to set the Subtype control as required when the displayed record is an active lead.

- Display an asterisk by the Subtype control for active leads
- Remove the asterisk by the Subtype control for other records (when Status or Type is changed)
- Save the msgError resource string as a page variable for use in the validation error message

- Configure a validation error message using the saved page variable
- Save all configurations and exit design mode

Displaying an asterisk by the Subtype control for display or update of active leads

You can dynamically indicate the requirement for Subtype when the OEP user displays or updates an active lead on an individual edit page by doing one of the following:

- Loading a pre-existing record that predates the Subtype requirement for active leads (Load event for the Page object)
- Changing the value for Status or Type (Change event for the Page object)

In this procedure, the action statement uses the Boolean AND operator (the AND operator is implicit); all conditions must be satisfied for the action to occur.

Each event uses the same action statement. In this procedure, you configure the statement once and then clone (copy) the statement to the other event.

Because this procedure uses the Load event, you must reload the page after completing the procedure to see the changes.

To display an asterisk by the Subtype control for display or update of an active lead:

1. In Dynamic Forms Designer ( ) for the individual edit page, add an action statement to the **Load** event for the **Page** object.
2. In the **Description** text box, type: *Add asterisk to Subtype if active lead*
3. Configure the conditions as follows:
   a. From the **Object** drop-down list, expand **Control**, expand **Status**, and then select **Get Value**.
   b. From the **Operator** drop-down list, select **Equal**; from the **Value** drop-down list, select **Active**.
   c. Click in the Summary section to add the condition to the action statement.
   d. From the **Object** drop-down list, expand **Control,** expand **Type**, and then select **Get Value**.
   e. From the **Operator** drop-down list, select **Equal**; from the **Value** drop-down list, select **Lead**.
   f. Click the top-most in the Summary section (above the "OR" operator) to add the condition to the action statement.
   g. Both conditions must be satisfied for the action to occur; the conditions have an implicit "AND" operator between them.
4. Configure the action as follows: From the **Object (Action)** drop-down list, expand **Control**, expand **Subtype**, and then select **Set Required (On)**.
5. Click to confirm changes.
6. Clone the statement and then move the cloned statement to the **Change** event for the **Page** object.

Removing the asterisk by the Subtype control for other records (when Status or Type is changed)

You can remove the asterisk by the Subtype control when the OEP user causes the record to become a record type other than "active lead" by changing the value in the Status control or the Type control.

In this procedure, the action statement uses the Boolean OR operator; only one of the conditions must be satisfied for the action to occur.

To remove the asterisk by the Subtype control when Status or Type is changed, resulting in a record that is not an active lead:

1. In Dynamic Forms Designer ( ) for the individual edit page, add an action statement to the **Change** event for the **Page** object.
2. In the **Description** text box, type: *Remove asterisk for Subtype if not active lead*
3. Configure the condition as follows:
   a. From the **Object** drop-down list, expand **Control**, expand **Status**, and then select **Get Value**.
   b. From the **Operator** drop-down list, select **Not Equal**; from the **Value** drop-down list, select **Active**.
   c. Click in the Summary section to add the condition to the action statement.
   d. From the **Object** drop-down list, expand **Control**, expand **Type**, and then select **Get Value**.
   e. From the **Operator** drop-down list, select **Not Equal**; from the **Value** drop-down list, select **Lead**.
   f. Click the bottom-most in the Summary section (below the "OR" operator) to add the condition to the action statement.

   Only one of the conditions must be satisfied for the action to occur, as indicated by the interceding "OR" operator.
4. Configure the action as follows: From the Object (Action) drop-down list, expand Control, expand Subtype, and then select Set Required (Off).
5. Click to confirm changes.

Saving a resource string as a page variable

You can save the msgError resource string as a page variable for use in the validation error message. (The msgError resource string is available for this page at installation.) When you save resource strings as page variables, they become available for events that occur after the selected event for the current session of the selected page. For example, if you save the msgError resource string as a page variable at the Load event for the individual edit page, then the saved page variable is available for the Validate event, which occurs after the Load event.

Note: For information on adding custom resource strings, see Using UI Text Editor.

To save the msgError resource string as a page variable:

1. In Dynamic Forms Designer ( ) for the individual edit page, add an action statement to the **Load** event for the **Page** object.
2. In the **Description** text box, type: *Save msgError as page variable*.
3. From the **Object (Action)** drop-down list, expand **Common** and then select **Get Resource String**.
4. Configure the source and value for **Resource File ID** as follows: From the **Source** drop-down list, select **User Defined**; in the **Value** text box, type: *individualedit*
5. Configure the source and value for **Resource Name** as follows: From the **Source** drop-down list, select **User Defined**; in the **Value** text box, type: *msgError*
6. Select the **Save Result?** check box; in the **Variable** text box, type: *msgError*
7. Click to confirm changes.

Configuring a validation error message using the saved page variable

You can configure a validation error message using the saved page variable from the msgError resource string (see earlier procedure Saving a resource string as a page variable).

In this procedure, the action statement uses the Boolean AND operator (the AND operator is implicit); all conditions must be satisfied for the action to occur.

To configure a validation error message using the saved page variable:

1. In Dynamic Forms Designer ( ) for the individual edit page, add an action statement to the **Validate** event for the **Page** object.
2. In the **Description** text box, type: *Display validation message if empty Subtype on active lead*.
3. Configure the conditions as follows (checks for active leads with empty Subtype):
   a. From the **Object** drop-down list, expand **Control**, expand **Status**, and then select **Get Value**.
   b. From the **Operator** drop-down list, select **Equal**; from the **Value** drop-down list, select **Active**.
   c. Click in the Summary section to add the condition to the action statement.
   d. From the **Object** drop-down list, expand **Control**, expand **Type**, and then select **Get Value**.
   e. From the **Operator** drop-down list, select **Equal**; from the **Value** drop-down list, select **Lead**.
   f. Click the top-most in the Summary section (above the "OR" operator) to add the condition to the action statement.

      Both conditions must be satisfied for the action to occur; the conditions have an implicit "AND" operator between them.

g. From the **Object** drop-down list, expand **Control**, expand **Sub_Type**, and then select **Get Value**.

h. From the **Operator** drop-down list, select Is **Empty**.

i. Click the top-most in the Summary section (above the "OR" operator) to add the condition to the action statement.

All conditions must be satisfied for the action to occur; the conditions have an implicit "AND" operator between them.

4. Configure the action as follows:

a. From the **Object (Action)** drop-down list, expand **Common**, expand **UI**, and then select **Show Validation Message Box**.

b. From the **Object (Action)** drop-down list, expand **Common**, expand **UI**, and then select **Show Validation Message Box**.

(Row)	For the Source, do this:	For the Value, do this:
Title	Select **Page Variable**	Select **msgError**
Message	Select **User Defined**	Type *Select a Subtype before saving an active lead.*
Icon Type	Select **User Defined**	Select **Exclamation**

c. Select the **Stop on Execute?** check box to prevent subsequent statements within the Validate event from executing if the conditions are met.

For more information on the Stop on Execute? check box, see Actions and conditions.

5. Click to confirm changes.

Saving all configurations and exiting design mode

When you complete all procedures in this topic, you must save the configurations.

To save all configurations and exit design mode:

- Close Dynamic Forms Designer and then click on the UCW toolbar to save all configurations and exit design mode.

Highlighting Controls

Using UCW, you can highlight controls in UCW-enabled pages without any programming knowledge.

This sample UI configuration demonstrates highlighting a control by configuring action statements that depend on the user-selected value for another set of controls.

Scenario used in this sample UI configuration

The manager of the remote sales team asks you to highlight the Subtype control when her staff adds or edits "active lead" records (on the individual edit page). Highlighted controls are outlined in red. This outlining helps the salespeople remember to select a hot, warm, or cold status for active leads.

The following graphic illustrates the Subtype control, highlighted.

Procedures in this sample UI configuration

These procedures use Dynamic Forms Designer. For basic instructions on using Dynamic Forms Designer, see Adding and modifying action statements.

Follow these procedures to highlight the Subtype control when the displayed record is an active lead.

- Highlight the Subtype control for active leads
- Remove the highlighting from the Subtype control for other records
- Save all configurations and exit design mode

Highlighting the Subtype control for display or update of active leads

You can dynamically highlight (outline) the Subtype control when the OEP user displays or updates an active lead on the individual edit page by doing one of the following:

- Loading a pre-existing record that predates the Subtype requirement for active leads (Load event for the Page object)
- Changing the value for Status or Type (Change event for the Page object)

In this procedure, the action statement uses the Boolean AND operator (the AND operator is implicit); all conditions must be satisfied for the action to occur.

Each event uses the same action statement. In this procedure, you configure the statement once and then clone (copy) the statement to the other related events.

Because this procedure uses the Load event, you must reload the page after completing the procedure to see the changes.

To highlight the Subtype control for display or update of an active lead:

1. In Dynamic Forms Designer ( ) for the individual edit page, add an action statement to the **Load** event for the **Page** object.
2. In the **Description** text box, type: *Highlight Subtype if active lead*
3. Configure the conditions as follows (checks for active leads):
   a. From the **Object** drop-down list, expand **Control**, expand **Status**, and then select **Get Value**.
   b. From the **Operator** drop-down list, select **Equal**; from the **Value** drop-down list, select **Active**.

c. Click in the Summary section to add the condition to the action statement.

d. From the **Object** drop-down list, expand **Control**, expand **Type**, and then select **Get Value**.

e. From the **Operator** drop-down list, select **Equal**; from the **Value** drop-down list, select **Lead**.

f. Click the top-most in the Summary section (above the "OR" operator) to add the condition to the action statement.

Both conditions must be satisfied for the action to occur; the conditions have an implicit "AND" operator between them.

4. Configure the action as follows: From the **Object (Action)** drop-down list, expand **Control**, expand **Subtype**, and then select **Highlight (On)**.
5. Click to confirm changes.
6. Clone the statement and then move the cloned statement to the Change event for the Page object.

Removing the highlighting from the Subtype control for other records

You can remove the highlighting from the Subtype control when the OEP user causes the record to become a record type other than "active lead" by changing the value in the Status control or the Type control.

In this procedure, the action statement uses the Boolean OR operator; only one of the conditions must be satisfied for the action to occur.

To remove the highlighting from the Subtype control when Status or Type is changed, resulting in a record that is not an active lead:

1. In Dynamic Forms Designer ( ) for the individual edit page, add an action statement to the **Change** event for the **Page** object.
2. In the **Description** text box, type: *Remove highlighting from Subtype if not active lead*
3. Configure the conditions as follows:

a. From the **Object** drop-down list, expand **Control,** expand **Status**, and then select **Get Value**.

b. From the **Operator** drop-down list, select **Not Equal**; from the **Value** drop-down list, select **Active**.

c. Click in the Summary section to add the condition to the action statement.

d. From the **Object** drop-down list, expand **Control**, expand **Type**, and then select **Get Value**.

e. From the **Operator** drop-down list, select **Not Equal**; from the **Value** drop-down list, select **Lead**.

f. Click the bottom-most in the Summary section (below the "OR" operator) to add the condition to the action statement.

Only one of the conditions must be satisfied for the action to occur, as indicated by the interceding "OR" operator.

4. Configure the action as follows: From the **Object (Action)** drop-down list, expand **Control**, expand **Subtype**, and then select **Highlight (Off)**.
5. Click  to confirm changes.

Saving all configurations and exiting design mode

When you complete all procedures in this topic, you must save the configurations.

To save all configurations and exit design mode:

- Close Dynamic Forms Designer and then click  on the UCW toolbar to save all configurations and exit design mode.

Changing Pages Dynamically

Using UCW, you can dynamically change UCW-enabled pages without any programming knowledge.

This sample UI configuration demonstrates hiding a tab and toolbar button by configuring action statements that depend on the user-selected value for another set of controls.

Scenario used in this sample UI configuration

The loan officers at your company want the Organization Chart tab (on the individual PowerPage) to be displayed only when needed, which is when the individual's title is specified in the individual record. The loan officers also want the Clone toolbar button to be available for active records only.

Procedures in this sample UI configuration

These procedures use Dynamic Forms Designer. For basic instructions on using Dynamic Forms Designer, see Adding and modifying action statements.

Follow these procedures to hide the Organization Chart tab when Title is empty and to disable the Clone toolbar button when Status is Inactive.

- Hide the Organization Chart tab if Title is empty
- Disable the Clone toolbar button if Status is Inactive
- Save all configurations and exit design mode

Hiding the Organization Chart tab when Title is empty

You can dynamically hide the Organization Chart tab when the record displayed on the individual PowerPage has no Title entry.

Because this procedure uses the Load event, you must reload the page after completing the procedure to see the changes.

To hide the Organization Chart tab when Title is empty (and show the tab when Title has a value):

1. In Dynamic Forms Designer ( ) for the individual PowerPage, add an action statement to the **Load** event for the **Page** object.
2. In the **Description** text box, type: *Hide Organization Chart tab if Title is empty*.
3. Configure the condition as follows (checks for empty Title):
   a. From the **Object** drop-down list, expand **Control**, expand **Individual_Title**, and then select **Get Value**.
   b. From the **Operator** drop-down list, select Is **Empty**.
   c. Click in the Summary section to add the condition to the action statement.
4. Configure the action as follows:
   a. From the **Object (Action)** drop-down list, expand **Control**, expand **Top_Tab**, and then select **Hide Item**.
   b. From the **Source** drop-down list, select **User Defined**; from the **Value** drop-down list, select **Organization Chart**.
5. Click to confirm changes.
6. Add another action statement to the **Load** event; in the **Description** text box, type: *Show Organization Chart tab if Title contains a value*.
7. Configure the condition as follows (checks for a Title value):
   a. From the **Object** drop-down list, expand **Control**, expand **Individual_Title**, and then select **Get Value**.
   b. From the **Operator** drop-down list, select **Is Not Empty**.
   c. Click in the Summary section to add the condition to the action statement.
8. Configure the action as follows:
   a. From the **Object (Action)** drop-down list, expand **Control**, expand **Top_Tab**, and then select **Show Item**.
   b. From the **Source** drop-down list, select **User Defined**; from the **Value** drop-down list, select **Organization Chart**.
9. Click to confirm changes.

Disabling the Clone toolbar button for inactive status records

You can dynamically disable the Clone toolbar button when the record displayed on the individual PowerPage has an inactive status. Disabled buttons are displayed with a dithered or "grayed-out"

appearance.

Because this procedure uses the Load event, you must reload the page after completing the procedure to see the changes.

To disable the Clone toolbar button for inactive status records:

1. In Dynamic Forms Designer ( ) for the individual PowerPage, add an action statement to the **Load** event for the **Page** object.
2. In the **Description** text box, type: *Disable Clone toolbar button for inactive status records*.
3. Configure the condition as follows (checks Status for Inactive value):
   a. From the **Object** drop-down list, expand **Control**, expand **Individual_Status**, and then select **Get Value**.
   b. From the **Operator** drop-down list, select **Equal**; from the **Value** drop-down list, select **Inactive** .
   c. Click in the Summary section to add the condition to the action statement.
4. Configure the action as follows:
   a. From the **Object (Action)** drop-down list, expand **Control**, expand **Top_Toolbar**, and then select **Disable Item**.
   b. Configure the **Item ID** source and value: From the **Source** drop-down list, select **User Defined**; from the **Value** drop-down list, select **Clone Individual**.
5. Click to confirm changes.
6. Add another action statement to the **Load** event; in the **Description** text box, type: *Enable Clone toolbar button if status is not inactive*.
7. Configure the condition as follows (checks for inactive status):
   a. From the **Object** drop-down list, expand **Control**, expand **Individual_Status**, and then select **Get Value**.
   b. From the **Operator** drop-down list, select **NotEqual**; from the **Value** drop-down list, select **Inactive** .
   c. Click in the Summary section to add the condition to the action statement.
8. Configure the action as follows:
   a. From the **Object (Action)** drop-down list, expand **Control**, expand **Top_Toolbar**, and then select **Enable Item**.
   b. Configure the **Item ID** source and value: From the **Source** drop-down list, select **User Defined**; from the **Value** drop-down list, select **Clone Individual**.
9. Click to confirm changes.

Saving all configurations and exiting design mode

When you complete all procedures in this topic, you must save the configurations.

To save all configurations and exit design mode:

- Close Dynamic Forms Designer and then click  on the UCW toolbar to save all configurations and exit design mode.

Advanced Configuration Examples

The advanced configurations in this book provide instructions for implementing the following UCW design mode configurations that require Advanced View coding.

Bundling Multiple Actions

This example demonstrates how to use Advanced View in Dynamic Forms Designer and the invokeAction function to bundle actions using an if…else statement. An action statement that bundles actions can execute multiple actions more efficiently. Also, as shown in this example, you can use Advanced View to incorporate an if …else statement that performs one of two actions based on a condition.

When entering or updating a company record and the Status is changed to “Hot,” highlight the Telephones section of the company edit page. When the status is changed to any other status value, highlight the URL box.

To implement this example:

1. In Dynamic Forms Designer for the company edit page, expand **Control**, expand **Status**, and then add an action statement to the **Change** event.
2. In the **Description** box at the top of the Action Designer pane, type a description of the action statement. Example description: "Bundle actions example."
3. Configure the condition as followings:
   a. In the **Object** drop-down list located in the Conditions section, expand **Control**, expand **Status**, and then select **Get Value**.
   b. From the **Operator** drop-down list, select **Equal**; and from the **Value** drop-down list, select **Hot**.
   c. Click  in the **Summary** section to add the condition to the action.
4. Configure the first action as following: in the **Object / Action** drop-down list, expand **Control**, expand **Telephones**, and then select **SetHighlight (On)**.
5. Click  to enable Advanced View and display the **Code** section.

Contents of the Code section

```
if ((invokeAction("indirect", "companyStatus", "getValue",
new Array("eventArgument1", psEventArgument1),
new Array("eventArgument2", psEventArgument2)).value == "313"))
{
  // Action Invoke ""setHighlightOn" on "telephones"
oUcfActionReturn = invokeAction("indirect", "telephones",
"setHighlightOn",
new Array("eventArgument1", psEventArgument1),
new Array("eventArgument2", psEventArgument2));
// terminate event if stop set, and return action object
if (oUcfActionReturn.stop) {
  return oUcfActionReturn;
}
}
```

6. On a line at the bottom of the Code section, type: else
7. Position the cursor below else, and then press CTRL+K to display the list of actions within the **Code** section.
8. Expand **Control**, expand **URL**, and then select **Set Highlight (On)**. This selection inserts the code for this action in the **Code** section.

Contents of the Code section

```
if ((invokeAction("indirect", "companyStatus", "getValue",
new Array("eventArgument1", psEventArgument1),
new Array("eventArgument2", psEventArgument2)).value == "313"))
{
  // Action Invoke ""setHighlightOn" on "telephones"
oUcfActionReturn = invokeAction("indirect", "telephones",
"setHighlightOn",
new Array("eventArgument1", psEventArgument1),
new Array("eventArgument2", psEventArgument2));
// terminate event if stop set, and return action object
if (oUcfActionReturn.stop) {
    return oUcfActionReturn;
```

```
        }
        }
        else
        invokeAction("indirect", "companyUrl", "setHighlightOn")
```

9. Click to save the action statement and then close the **Dynamic Forms Designer** window.
10. Click on the UCW toolbar to save all configurations and exit design mode.

To test this example:

1. Display a record in the OEP company edit page.
2. Change the contents of the Status value to either Warm or Cold.

   Check that the Url text box is highlighted.
3. Save the record.
4. Redisplay the record in the company edit page.
5. Change the Status value to Hot.

   Check that the Telephones section is highlighted.

Applying Custom Styles to a Page

This example applies HTML element styles to the company edit page using ASP Editor Dynamic Forms Designer. The styles put a border around the element identified as "SectionDetailsMain."

Onyx does not recommend that you modify the underlying HTML of a UCW-enabled area; therefore, do not create or modify element identifiers or classes. You can add styles and override existing styles of elements by typing specifications directly into the ASP editor.

To implement this example:

1. In Dynamic Forms Designer for the company edit page, expand **Server** and then expand **Body**.

   The Before and After events are displayed in the Events pane. These events indicate available insertion points relative to the BODY element of the company edit ASP page.
2. Select the **Before** event and then click to display the ASP/HTML section on the right of the window.
3. In the **Description** box above the ASP/HTML section, type a description of the ASP statement.
4. In the **ASP/HTML** section, type the following:

```
        <style>
```

```
#SectionDetailsMain{border-top: 3px solid red; border-bottom:
3px solid red; border-left: 1px solid gray; border-right: 1px
solid gray; padding: 5px;}
</style>
```

5. Click to save the ASP statement and then close the Dynamic Forms Designer window.
6. Click on the UCW toolbar to save all configurations and exit design mode.
7. To test the page appearance, load the individual edit page. If the changes do not appear, clear the Temporary Internet files from the client that you are using and reload the page.

Applying a Custom CSS to a Page

UCW-enabled OEP pages accept custom cascading style sheet (CSS) files by specifying link statements using ASP Editor in the Dynamic Forms Designer. This example applies borders to the sections of the individual edit page. The OEP product CD contains the referenced CSS file in the Examples directory.

Onyx recommends that you avoid modifying the underlying HTML of a UCW-enabled page; therefore, do not create or modify element identifiers or classes. You can add styles and override existing styles of elements with custom CSSs.

To implement this example:

1. Copy the custom.css file located on the OEP product CD in the ..\Documentation\Technical Guide\Examples\StyleSheet directory to the YourOEPwebsite\ucf\data\custom\{your company name} directory in your development environment.
2. In Dynamic Forms Designer for the individual edit page, expand **Server** and then expand **Body**.

   The Before and After events are displayed in the Events pane. These events indicate available insertion points relative to the BODY element of the company edit ASP page.

3. Select the **Before** event and then click to display the ASP/HTML section on the right of the window.
4. In the **Description** text box above the ASP/HTML section, type a description of the ASP statement.
5. In the **ASP/HTML** section, type the following link statement:

```
<link rel="stylesheet" type="text/css" href="..\ucf\data\custom\
{your company name}\custom.css">
```

6. Click to save the ASP statement and then close Dynamic Forms Designer.
7. Click on the UCW toolbar to save all configurations and exit design mode.

8. To test the page appearance, load the individual edit page. If the changes do not appear, clear the temporary Internet files from the client that you are using and reload the page.

Adding a Date/time Control

This example adds a textbox control to the company edit page that displays the date and time that the company record was last modified. Because the updateDate is stored in the database in UTC format, Advanced View code is required to convert the updateDate to the local time at the OEP client.

Note: The toolbox date control and the intrinsic simple date control (these controls display a date only; they do not display a time) do not perform time zone conversions.

To implement this example:

1. Display the company edit page and enter design mode.
2. Add two textbox toolbox controls to the Details section of the page. Position the controls under the DUNS text box.
3. Configure one of the control's properties using the Properties entries shown in the following table. Typed entries are indicated by italics; selected entries are indicated by boldface. For property settings not listed in the tables, accept their default values (which can be empty).

Property	Entry
Admin ID	*Last_Update_Display*
Tooltip Text	*Displays the date and time of the last update*
Label	*Last Update*
Mode	**Read-only**

4. Configure the other textbox control bound to company.updateDate using the following Properties entries.

Property	Entry
Admin ID	*Last_Update_Hide*
Object	**Company**
Property	**updateDate**
Mode	**Read-only**

5. Add an unconditional action statement on the **Load** event that hides the Last_Update_Hide control. For more information, see Hiding section and controls.

6. Add an unconditional action on the **Load** event. This action consists of Advanced View code the retrieves the updateDate from the hidden control, converts it from UTC to the local time at the OEP client, and sets the displayed control to the converted date and time.
7. In the **Description** box above the Code section, type a description of the action.
8. Click  to enable Advanced View.
9. Put the following sample JavaScript code into the **Code** section.

```
try{
  // This is the displayed control with the local dateTime
 var sIdVisibleControl = 'uid_1153519424601_6106_34__id34';
  // This is the hidden control with the UTC dateTime
  var sIdHiddenControl = 'uid_1153519339711_3104_28__id21';
 // Get the value of the hidden control...
  var sValue = invokeAction('indirect', sIdHiddenControl,
'getValue').value;
 // ...convert it to local time if it's not blank...
  if (sValue != '') {
   var oXDatetime = new XDatetime();
   sValue = oXDatetime.convertUTCToLocal(sValue, DATEFORMAT_
DATETIME);
   if(sValue < 0) sValue = '';
  }
 // ...and store it in the visible control
  invokeAction('indirect', sIdVisibleControl, 'setValue',
   new Array('value', sValue));
}
catch(oError){
// display error message - unable to load datetime
}
```

10. Obtain the IDs (uid_1153864612372_3552_28__id58, for example) for the two textbox controls that you specified previously. In the Code section, position the cursor below the code entered in the previous step. Use the Ctrl-K keystroke to display the object/action list in the Code section. From the object action list select **Control** and then **Last_Update_Display**. Select any action from the list. The code for the selected action displays, and the ID is the second argument in the

invokeAction call. Display action code for the Last_Update_Hide control in the same manner.

11. Cut and paste the IDs into the code getValue and setValue invokeAction calls. The code comments indicate where to paste the IDs. Delete the code that you used to obtain the control ID.
12. Click  to save the action.
13. Add an unconditional action on the **Save** event. This action consists of Advanced View code the retrieves the updateDate from the displayed control, converts it from the local time to UTC, and sets the displayed control to the converted date and time.
14. Click  to enable Advanced View.
15. Put the following sample JavaScript code into the **Code** section.

```
try{
  var oXDatetime = newXDatetime();
  // This is the displayed control with the local dateTime
  var sValue = invokeAction('indirect', 'uid_1153519424601_6106_
34__id34', 'getValue');
 sValue = oXDatetime.convertLocalToUTC(sValue, DATEFORMAT_
DATETIME);
  if(sValue < 0) sValue = '';
 // This is the hidden control with the UTC dateTime
 invokeAction('indirect', 'uid_1153519339711_3104_28__id21',
'setValue',
   new Array('value', sValue));
}
catch(oError){
// display error message - unable to persist datetime
}
```

16. Click  to save the action.
17. Close the Dynamic Forms Designer.
18. Click  on the UCW toolbar to save all configurations and exit design mode.

Filtering Domain Data by Profile

This example filters the Source control on the incident edit page. The filtering is done using Advanced View in Dynamic Forms Designer. This code removes one or more values from the Source dropdown

list on the incident page. With the filter, OEP displays a subset of Source values to choose from.

Scenario used in this example UI configuration

The primary task of the telephone agents in a company is to field inbound calls and email messages and create customer and incident records resulting from those inquiries. The calls are the result of a marketing campaign that provides potential customers with an 800 number and an email address for contacting the company. Therefore, the only valid incident Source values for these telephone agents are “Phone” and “Email.” This example removes “Mail” and “Fax” from the drop-down list.

> **Important:** When planning configurations that affect the domain data that OEP users can select, consider what problems may arise. For instance, determine how to handle or avoid the situation where an OEP user attempts to update a record that contains domain data values that do not appear in the user's drop-down lists. This example does not contain code to mitigate such problems.

To implement this example:

1. Display the PowerPage for a customer, select the **Sales** tab, and select to display the incident edit page. Enter design mode for the incident edit page.
2. Drag the **Source** UI control from the page to the Intrinsic Controls pane.
3. Drag the **Dropdown** toolbox control from the Toolbox window to the location previously occupied by the Source control.
4. Configure the control's properties using the following entries.

Property	Entry
Admin ID	*Filtered_Source*
Tooltip Text	*Displays filtered source list*
Label	*Source*
Field Name	*incident.source*
Parent ID	*3*
Property	*sourceId*
Mode	**Read/Write**

This configuration information binds the dropdown control to the sourceId property of the incident object when incidents are sales opportunities (Parent ID equals 3).

5. Click on the UCW toolbar to save all configurations and exit design mode.
6. In Dynamic Forms Designer for the incident edit page, add an unconditional action statement to the **Load** event for the **Page** object. This action is used temporarily to obtain XML used by OEP to populate the Source control.

7. In the **Description** box above the Code section, type a description of the action.
8. Click  to enable Advanced View.
9. Put the following sample JavaScript code into the **Code** section.

```
// Call the getValueTextCollection action to get Source items
for Filtered_Source.
// Note that the second parameter, the identifier of the control

// (uid_1153760193605_1921_55__id23 is unique and must be
obtained from your
// implementation.
sourceData = invokeAction("indirect", "uid_1153760193605_1921_
55__id23",
   "getValueTextCollection");
// Display the contents of sourceData in an alert message.
alert(sourceData);
```

10. Obtain the ID for the Dropdown toolbox control that you specified previously. In the Code section, position the cursor below the code entered in the previous step. Use the Ctrl-K keystroke to display the object/action list in the Code section. From the object action list, select **Control** and then **Filtered_Source**, which is the Admin ID of the Dropdown toolbox control. Select any action from the list. If you chose the Get Value action, for example, the following code appears:

```
/* Get value from 'Filtered_Source' */ invokeAction("indirect",
"uid_1153346314029_4496_49__id83", "getValue")
```

    Cut and paste the identifier of the Filtered_Source control into the getValueTextCollection call. Delete the code that you used to obtain the control ID.

11. With the cursor in the Code section, press Ctrl-D to enter developer mode. Press Ctrl-I to execute the code, which opens an alert window that displays the contents of the sourceData variable. This variable contains XML that defines the dropdown list items of the Source control. The XML displayed in the alert window is:

```
<items sortType="numeric" sortDirection="ascending">
 <item value="" text="" sequence="1"/>
 <item value="126" text="Mail" sequence="2"/>
 <item value="132" text="Phone" sequence="3"/>
 <item value="114" text="E-mail" sequence="4"/>
```

```
      <item value="115" text="Fax" sequence="5"/>
    </items>
```

12. Record the XML and either disable or delete the action that you used to obtain the Source XML.
13. Add an action statement to the **Load** event for the **Page** object.
14. Configure the action as follows:

    a. From the **Object (Action)** drop-down list, expand **Control**, expand **Filtered_Source**, and then select **Set Reference Data**.

    b. From the **Source** drop-down list for **Reference Data XML**, select **User Defined**; in the **Value** box, enter the XML string for Source with the Fax and Mail item elements removed. Also remove the first item element, which has an empty value attribute. Renumber the sequence attribute for the remaining Phone and E-mail item elements. If necessary, remove line breaks in the XML. The XML string to enter in the **Value** box is:

```
<items sortType="numeric" sortDirection="ascending"><item
value="132"
text="Phone" sequence="1"/><item value="114" text="E-mail"
sequence="2"/></items>
```

    c. From the Source drop-down list for Reselect Value, select User Defined; from the Value drop-down list, select Yes to force the control to reselect its persisted value (if any) from the newly set reference data. If the persisted value is not defined in the new reference data, the control displays an empty control. Selecting No removes any persisted value from the control so that the OEP user can select from the new reference data.

15. Click  to save the action.
16. To prevent validation errors, deactivate the following default Onyx action statements, which reference the removed default Source control.

    - Load object:
      - Onyx: Set Source Required
      - Onyx: Specify Source Get Resource for Validation
    - Validate object:
      - Onyx: Source (Required)

17. Close the Dynamic Forms Designer.
18. Click  on the UCW toolbar to save all configurations and exit design mode.

Adding an HTML Container Control

This example adds a tab to the individual PowerPage that displays a domain lookup page from an external website. The domain lookup string is derived from the contents of the Email text box control. Using an HTML Container toolbox control, this example demonstrates how to give users one-click access to information. It shows how quickly external pages or custom OEP pages can be configured for display from the tabs on OEP pages.

To implement this example:

1. Enter design mode for the individual PowerPage.
2. Use Navigation Designer to add a custom bottom tab to the individual PowerPage. Save the tab with the following properties.

ID	Caption (name)
emailLookup	Email Lookup

   For basic instructions on adding custom tabs, see Configuring tabs.

3. Drag an HTML Container toolbox control from the Toolbox window to the custom tab's panel corner ( ).

   For basic instructions on adding custom controls to a page, see Adding and moving controls.

4. Write the control's ID property (first property in the Properties window) to a file. Example ID: "uid_1147926514330_4762_88__id22"

   The control ID identifies the HTML Container toolbox control added in the previous step. Use the control ID as the second argument for the getUiContainer method, which is called by the code used in the next step.

5. Add an Load event to the individual PowerPage using Advanced View that contains the code below. If you cut and paste this code in Advanced View, you may need to make some minor edits, such as removing extra line breaks, to prevent syntax errors.

Contents of the code section

```
// This is a conditional statement that insures the domain
lookup is attempted
// only when the individualEmail control is not empty.


if ((/*Get value from 'Individual_Email'*/ invokeAction
("indirect", "individualEmail",
    "getValue", new Array("eventArgument1", psEventArgument1),
    new Array("eventArgument2", psEventArgument2)).value !=
""))
```

```
// Retrieve the email string from the individualEmail control
and save it to a variable

{

var individualEmail = invokeAction("indirect",
"individualEmail", "getValue",

    new Array("eventArgument1", psEventArgument1), new Array
("eventArgument2",

    psEventArgument2)).value;


// Get a reference to the HTML container control using the
control ID that was noted when

// the HTML container control has added to the individual
Powerpage


var oContainer = moUcfFrameworkRuntime.getUiContainer
("indirect","uid_1153346314029_4496_49__id83" );

  if (oContainer) {

   // Remove the user name and @ from the email address

   var domainStart = individualEmail.substring(0,
individualEmail.lastIndexOf('@')+1);

   var domainString = individualEmail.substring
(domainStart.length, individualEmail.length+1);

   // Display the domain lookup web page

   oContainer.innerHTML = "<html><body
style='width:100%;height:100%'>

    <iframe id='webPageHolder'

    src='http://registrar.verisign-grs.com/cgi-bin/whois?whois_
nic=" + domainString + "'

    style='width:100%;height:100%' ></iframe></body></html>";

  }

}
```

6. Click  to save the action statement and then close Dynamic Forms Designer.
7. Click  on the UCW toolbar to save all configurations and exit design mode.

Adding a Data-bound HTML Container Control

This example adds a HTML container toolbox control to the company edit page that is bound to the dunsNumber property of the company object. The HTML container control lets you specify custom formatting for a control. In this case. for demonstration purposes, it specifies a text box with a red background and a white font for displaying the control's value. In order to create a DUNS control with alternate formatting, you must remove the intrinsic, or default, DUNS control and replace it with an HTML container control.

A data-bound HTML container control requires you to create handlers using Advanced View code that enable set value and get value functionality, which is required for writing and retrieving values that the database persists. The JavaScript code in the following procedure demonstrates this functionality.

To implement this example:

1. Display the PowerPage for a company, and click  to display the company edit page. Enter design mode.
2. Drag the **D-U-N-S** UI control from the page to the **Intrinsic Controls pane.**
3. Drag the **HTML Container** toolbox control from the Toolbox pane to the location previously occupied by the D-U-N-S control.
4. Configure the control's properties using the following entries in the Properties pane. Typed entries are indicated by italics; selected entries are indicated by boldface. For property settings not listed in the tables, accept their default values (which can be empty).

Property	Entry
Admin ID	*Duns_Custom*
Property	**dunsNumber**
Mode	**Read/Write**

This configuration information binds the dropdown control to the dunsNumber property of the company object.

5. In Dynamic Forms Designer for the company edit page, add an unconditional action statement to the **Load** event for the **Page** object. This action contains the code that enables the HTML container control.
6. Click  to enable Advanced View.
7. In the **Description** box at the top of the Action Designer pane, type a description of the action statement. Example description: "Load custom DUNS control."
8. Put the following sample JavaScript code into the **Code** section. This code enables the HTML container control.

JavaScript code

```
/* Create / Set Page Variable "myToolboxControl" equal to "uid_
```

```
1153499472681_1375_63__id29" */
invokeAction("direct", "ucf.common", "setPageVariable", new
Array("name", "myToolboxControl"),
     new Array("value", "uid_1153499472681_1375_63__id29"));


function html_getValue() {
// Get the toolbox ID from the myToolboxContol page variable
 var sHtmlContainerId = invokeAction("direct",
"ucf.variable:myToolboxControl", "getValue").value;
  var oInput = document.getElementById(sHtmlContainerId +
"input");
 if (oInput) {
    return new UcfActionReturn(oInput.value);
  } else {
   return new UcfActionReturn("", false);
      }
  }
function html_setValue(psValue) {
// Get the toolbox ID from the myToolboxContol page variable
 var sHtmlContainerId = invokeAction("direct",
"ucf.variable:myToolboxControl", "getValue").value;
 var oInput = document.getElementById(sHtmlContainerId +
"input");
  if (oInput) {
   oInput.value = psValue;
  }
   return new UcfActionReturn(true);
  }
// Get the toolbox ID from the myToolboxContol page variable
  var sHtmlContainerId = invokeAction("direct",
"ucf.variable:myToolboxControl", "getValue").value;
 var oContainer = moUcfFrameworkRuntime.getUiContainer
("indirect", sHtmlContainerId);
// Retrieve the current tab index value that UCW maintains
```

```
 var sTabIndex = oContainer.getAttribute("ucfTabIndex");


 if (oContainer) {
  // Create a red textbox control with a label. Note that the
tab index
  // is set for the control using the contents of sTabIndex.
  oContainer.setAttribute("ucfControlRef", sHtmlContainerId +
"input");
  oContainer.innerHTML = "<span id='" + sHtmlContainerId +
"label'  class='ucfLabel'>D-U-N-S</span><input id='" +
sHtmlContainerId + "input' type='text'
style='width:100%;background-color:red;color:white;font-
weight:bold;' tabindex='"+sTabIndex+"'/>";
  var sWidth = getUcfLabelWidth(sHtmlContainerId);
   var oLabel = document.getElementById(sHtmlContainerId +
"label");
  oLabel.style.width = sWidth ;
}
//Enable the setValue and getValue functions using these
internal variables
invokeAction("indirect", sHtmlContainerId, "setVariable", new
Array("id", "_getValueHandler"), new Array("value", html_
getValue));
invokeAction("indirect", sHtmlContainerId, "setVariable", new
Array("id", "_setValueHandler"), new Array("value", html_
setValue));
```

The preceding code demonstrates these items:

- Invoking the html_getValue() and html_setValue() functions by setting the contents of the_ setValueHandler and _getValueHandler internal variables to those functions.
- Obtaining a reference to the HTML container object using the UcfFrameworkRuntime object. The ID string stored in the myToolboxControl variable is found in the Properties pane when OEP is in design mode and the HTML container control is selected.
- Setting the contents of the HTML container, including a label for the textbox, using the innerHTML property of the container.
- Setting the tab index for the HTML container by retrieving the current ucfTabIndex value and using it to set the tabindex attribute of the textbox that is specified in the innerHTML.

- Determining the area required for the control (when a page resize event occurs) using the getUcfLabelWidth utility function from the following file: OEPebsite\ucf\infrastructure\page_manager\common_utility_functions.js.

9. Click  to save the action.
10. Click  to move the action above the "Onyx: Load" legacy action. Because the Onyx load action requires the DUNS custom control's setValue handler in order to display the data-bound property in the custom DUNS control, the custom DUNS control action must load prior to the Onyx load action.
11. Close the Dynamic Forms Designer.
12. Click  on the UCW toolbar to save all configurations and exit design mode.

About the HTML Container Control

The HTML container control is a Dynamic Forms Designer toolbox control that enables UI designers to create a custom control that contains HTML that you provide. Adding the HTML container control from the Toolbox creates a placeholder for the control ( ), but to render it requires custom code entered manually through Advanced View in Dynamic Forms Designer. Designers can use the HTML container control to create controls that are not provided in Dynamic Forms Designer. For example, this control can display custom HTML that hosts external pages or ActiveX controls.

There is no limit to the number of HTML container controls that can exist on a page, but they cannot be copied using the clone functionality of the Dynamic Forms Designer. Each control must be created separately.

The HTML container control has two internal variables (_setValueHandler and _getValueHandler) that allow Dynamic Forms Designer custom code to set handler functions for setValue and getValue actions. A special action, fireChange, enables custom code to force a change event to be invoked.

 **Note:** For more information about _setValueHandler and _getValueHandler, see Adding a data-bound HTML container control.

Dynamic Forms Designer Reference

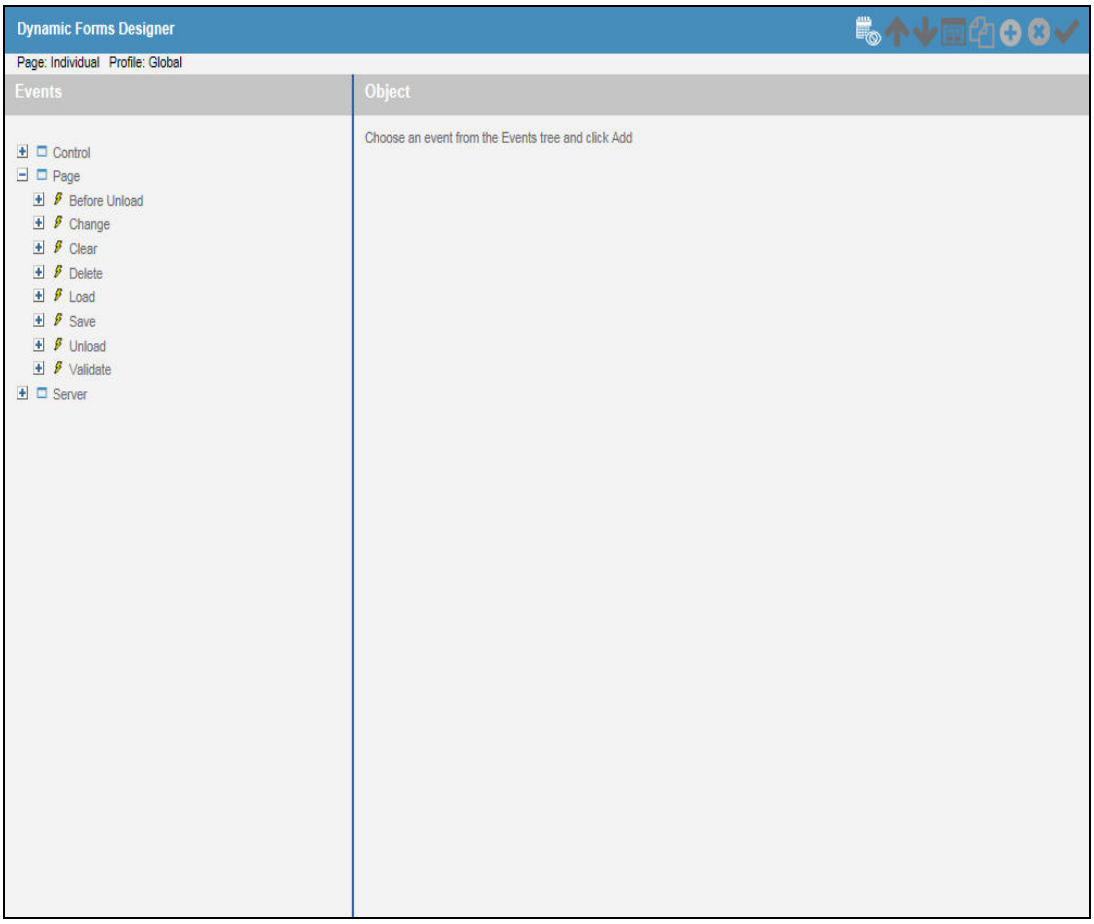
This book contains reference information for defining actions and conditions associated with OEP page events. Actions and conditions are configured using the UCW tool Dynamic Forms Designer. For information about using Dynamic Forms Designer, see Configuring page behavior.

Events

This section contains the events that appear in the Dynamic Forms Designer event panel. The events are grouped into objects so that events associated with specific UI controls are separate from events that pertain to the page as a whole. The event list differs for each UCW-enabled page. The following table lists the event objects.

Object	Description
Control	Identifies events associated with UI controls, usually as a result of user interaction with UI elements on a page.
Page	Identifies events associated with the page as a whole, such as loading of a page or validating user input.
Server	Identifies server-side events associated with the page.

The following graphic of the Events tree in Dynamic Forms Designer shows the event objects for the company PowerPage.



Control Events

Control objects, which appear in the Events pane of the Dynamic Forms Designer window, represent client-side events for UI elements, such as controls (including activated UDFs), tabs, toolbars, and, for the main application frame, Header Bar buttons and Navigation Bar links. Controls that UCW designers add to a page will also appear in this list.

In the Events pane, controls and custom toolbars are identified by their Admin ID properties while default toolbars, default tabs, and custom tabs are contained within toolbar or tab groups, such as "Top_Tab." For more information about the Admin ID property, see the Control properties reference.

Control events occur when OEP users update values associated with UI controls or when they select toolbar or tab items. On the other hand, a change event does not fire when Dynamic Forms action statements set the value of controls.

You can configure action statements for controls that have associated events. You cannot configure action statements for controls that do not have associated events. Such controls are display-only. OEP users cannot select or update values associated with display-only controls. The company and individual PowerPages contain many display-only controls. The company edit and individual edit pages contain many user-interactive controls.

The following table lists the control events:

Event	Available for these control objects	Occurs when an OEP user...
Change	UI controls, such as drop-down lists and check boxes	Changes the contents or selection of a UI control.
Item Click *(Item ID)*	Tabs	Clicks the tab identified by the Item ID.
Item Click *(Item ID) (Item Value)*	Toolbars	Clicks the toolbar item identified by the Item ID and Item Value.
Pause	Elapsed time pause button on the incident edit and task edit pages only	Clicks the elapsed time pause button on the incident edit and task edit pages.
Start	Elapsed time start button on the incident edit and task edit pages only	Clicks the elapsed time start button on the incident edit and task edit pages.

Page Events

The Page object, which appears in the Events pane of the Dynamic Forms Designer window, represents page client-side events for the current context (selected page or the main application frame). You can add action statements to the events listed for the Page object. Default Onyx action statements are prefixed with "Onyx."

The following table lists the page event objects.

Event	Occurs...	Event
Before Unload	When a user attempts to navigate away from page or close the browser window. This event lets you create action statements that warn the user of the consequences of closing the page, for example, that unsaved changes will be lost. If action statements under a Before Unload event use the Terminate and Return String action, the specified string is displayed in the Internet Explorer "Are you sure you want to navigate away from this page?" alert displayed to the user. Note that OEP	Before Unload

Event	Occurs...	Event
	automatically warns the user if the user made changes to standard controls on the page. Because the user may choose to cancel the navigation and stay at the current page, the fact that the beforeUnload event has fired does not necessarily mean that the page is about to be unloaded. Any actions to be executed only when the page is definitely about to be unloaded should be placed under the Unload Event instead.	
Change	When a change event occurs for any of the page's controls.	Change
Clear	When data is removed from the page.	Clear
Delete	When the user deletes the record that the page displays.	Delete
Load	When a browser loads the page.	Load
Resize	When a page loads and when an OEP user resizes, maximizes, minimizes, and restores a page. This event resizes the division markers ( ) on the page. Because resize can fire many times, do not attach significant actions (such as displaying a dialog box) to the resize event.	Resize
Retrieve	When the page uses an Onyx Enterprise Application Server (OEAS) method call to retrieve data.	Retrieve
Save	When an OEP user saves the page (assuming that all required validation is completed). The page uses an Onyx Enterprise Application Server (OEAS) method call to insert or update data.	Save
Unload	When the page is unloaded from the browser.	Unload
Validate	When the user attempts to save the page; before a save. The browser performs client-side validation actions at this event. Dynamic Forms Designer actions can prevent pages from saving when a conditional action statement used for validation indicates that the user entered invalid data or did not supply required information. In this case use the Show Validation Message Box action to alert the user and set the action statement return value to false (this prevents the save). You can also prevent a save by using the Terminate Current Event (Return Boolean) action that returns a value of false. For an example of using the Show Validation Message Box action, see Configuring a validation error message.	Validate

Server Object

The Server object appears in the Events pane of the Dynamic Forms Designer window. Body is the only object under Server, and you can add an ASP statement to the this object that insert server-side ASP code either before or after the <body> section of an ASP page. For more information, see Configuring page appearance (ASP Editor).

Actions and Conditions

Using Dynamic Forms Designer to write action statements alters OEP behavior dynamically as users interact with UCW-enabled OEP areas. Action statements can include conditions, which if true allow an action to execute.

The action and condition lists are organized by objects that represent UI elements, data for the currently displayed record, click event arguments, and so on. As you expand each object's node(s), the tree exposes the possible actions available for each object.

Conditions

Conditions are "get value" actions that compare the values to user-supplied expressions using operators available in a drop-down list. The Object/Action drop-down list in the Action section of the window contains many more actions than are available in the Object drop-down list in the Conditions section.

Arguments

When you select an action from the Object/Action drop-down list, any related arguments appear. Not all actions have arguments. Arguments are additional pieces of information that may or may not be required for an action to be successful. Each argument includes a source and a value. The source is selected from a drop-down list:

User Defined: Selecting this source displays a list of values (if the argument is limited to a list of values) or displays a text box for your typed entry. For example, user-defined values for the Item ID argument for the Top_Tab (Select Item) action are limited to the top tabs for the current page; user-defined values for the Title argument for the UI (Show Message Box) action are not limited, so a text box is displayed for your typed entry.

Page Variable: Selecting this source displays a list of page variables that contain values for the current area.

Argument data types are always string. Some actions use arguments that behave as Boolean data types, such as Set Recall and Set User Reminder; however, the data type is string.

Additional action settings (Save Result? and Stop on Execute?)

When you select an action from the action list, the Save Result? check box and the Stop on Execute? check box appear in the Action section of the Dynamic Forms Designer window.

The Save Result? check box stores the result of this statement as a page variable with the specified name. (The name is specified in the text box to the right of the check box, and it must be a valid JScript variable name.) Previously stored page variables are listed within the Page Variable source for action arguments and within the Variables object for conditions.

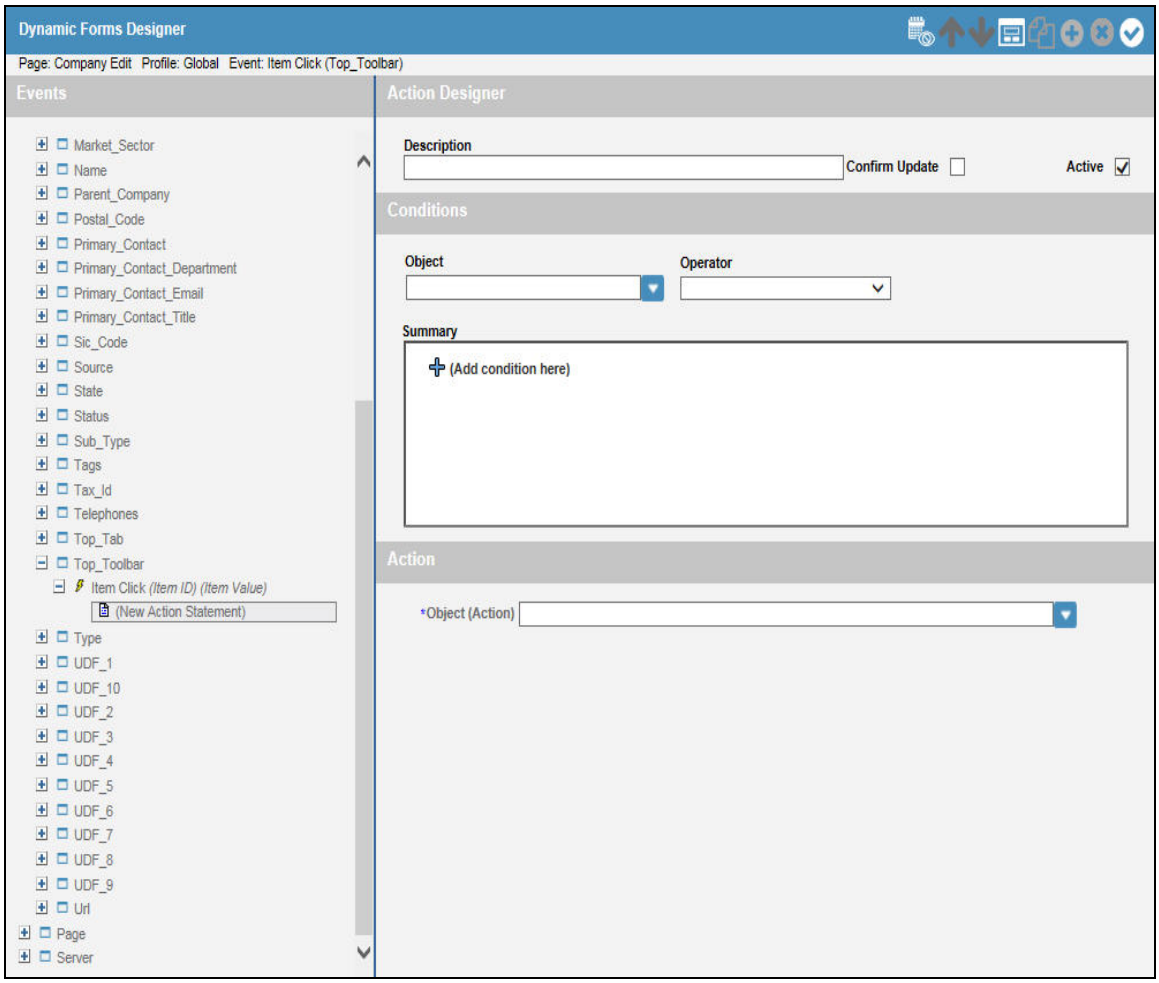
The Stop on Execute? check box stops events after specific statements (such as stopping the save event if a statement returns a true condition, which indicates an error) and skipping statements that have the same purpose (such as validating data). For example, the Validate event for a particular page may contain five statements that validate entered data, with each statement executing if a true condition is returned for a particular entry. (In this example, a true condition indicates that the related entry is invalid.) The final value for validation events is the last returned value. If the last statement

returns a false condition, the event is set to false, whether or not the entered data is valid. To prevent an erroneously false value for the validation event, you can end the event if any of its statements return a true condition.

Note: Selecting Stop on Execute? does not affect the returned value. If you want to stop the event and set the returned value, you can use one of the Terminate Current Event actions, located under the Common object in the list of actions.

Example graphic of action and condition lists

The following graphic of Dynamic Forms Designer for the company PowerPage shows the condition list (Object drop-down list in the Conditions section) and the action list (Object/Action drop-down list in the Action section).



Items returned by actions

Actions that retrieve data return the data as strings or integers. Actions that set values, evaluate expressions, display message boxes, show/hide controls, and so forth, return a Boolean value to indicate success or failure.

Objects in the action and condition lists

The following table lists the objects that appear in the lists of actions and conditions.

Object	Description of contents
Caption	Default caption controls for the current OEP area. Captions are identified by their Admin ID properties.
Common	Objects and actions available to all UCW-enabled areas, such as actions that display messages.
Context Data	Data available to the current OEP area, such as the parent company ID available to the company edit page.
Control	Actions available to UI controls elements, including custom caption controls, for the current context (page or main application frame). Controls that you move to the Intrinsic Controls pane also are listed with the available actions.
Event Arguments	Actions or conditions that retrieve the value of the argument of the Item Click event that causes the action statement to occur.
Page Variable	Page variables defined for the current OEP area.
Page	Actions or conditions that apply to the current context (page or main application frame).
Section	Actions for sections defined for the current OEP page. (This object does not appear in the condition list.)

Caption Actions and Conditions

Caption actions let designers manipulate strings for default captions that identify divisions on pages, such as "Billing Address" on the PowerPage. For conditions, you can use Get Caption Text to evaluate caption strings.

Captions are identified in the action and condition lists by their Admin ID properties.

Toolbox captions are enumerated by the Control object in the actions and conditions object lists.

Action	Description
Get Caption Text	Retrieves the text of a UI container caption
Hide Caption	Prevents a caption from displaying
Set Caption Text	Establishes or updates caption text
Show Caption	Displays caption text

Common Actions

The Common object consists of actions that are available to all UCW-enabled areas. Many of the actions in the Common object are defined in four sub-objects, and the remaining actions appear below those subobjects.

The common object contains the objects listed in the following table.

Object	Description
UI	Actions that display messages and windows to OEP users.
Navigation	Actions that display OEP feature pages to OEP users.
Data	Actions that retrieve XML data from OEAS and cached delta merge output, and actions that manipulate strings.
Debug	Actions that assist in debugging Dynamic Forms scripts, particular those developed using Advanced View.

The common object contains the actions listed in the following table.

Action	Description
Execute (Onyx) Legacy Code	Executes the default Onyx function call identified by the specified process ID.
Execute XML HTTP Call	Executes an XMLHTTP call to the specified URL, using the specified XML as input, and returns the result as a string.
Fire Event	Fires a Dynamic Forms event.
Get Current User Name	Retrieves the first and last name of the current OEP user.
Get Current User ID	Retrieves the user ID of the current OEP user.
Get Global Variable	Retrieves the value stored by the specified global variable.
Get Page Variable	Retrieves the value of the specified page variable. Page variables are available to actions and conditions on the current context only (page or main application frame).
Get Profile ID	Retrieves the OEP profile ID of the current user.
Get Profile Name	Retrieves the OEP profile name of the current user.
Get Resource String	Retrieves the specified resource from the specified file. You can optionally specify up to two tokens to be inserted in the string. Only resource strings for the current context only (page or main application frame) can be obtained using this action.
Get User Name From ID	Retrieve the first and last name associated with the specified user ID.
Has Permission?	Check whether the current OEP user has permission to access the specified resource.
Set Global Variable	Set the specified global variable to the specified value.
Set Page	Set the specified page variable to the specified value. Page variables are available to

Action	Description
Variable	actions and conditions on the current context only (page or main application frame) only.
Terminate Current Event (Return Boolean)	Stops the execution of the event context and sets stop for the existing handler execution; returns a Boolean (true or false).
Terminate Current Event (Return String)	Stops the execution of the event context and sets stop for the existing handler execution; returns a string.
Validate Value	Validates a specified value. Value types are dateTime, date, and time strings. Also validates integers and floats.

UI Actions

UI actions display messages and windows. The UI object is listed in the action list only, under the Common object.

Action	Description
Open New Window	Opens a new browser window using the provided URL, target, and style parameters. The target argument specifies the ID of the window or frame where the URL is to be loaded. It is typically desirable to load the URL in a new window. In this case specify the target argument as "_new".
Show Message Box	Displays a message box.
Show Prompt Box	Displays a prompt box.
Show Validation Message Box	Displays a validation message box when validation criteria are not met. This action always returns false; the page cannot be saved. Calls to Show Validation Message Box typically use the Stop on Execute flag, to make sure that other condition statements do not alter the return value of the action.

Show message box, show prompt box, and show validation message box have the following parameters:

Title: a string that displays in the title bar of the box.

MessageText: a string that displays in the box.

Icon: image displayed in the box (critical = 1, question = 2, exclamation = 3, or information = 4).

ReplaceText~1: a string that replaces "~1" in the message text string.

Replace Text ~2: a string that replaces "~2" in the message text string.

Show prompt box has the following parameters, in addition the ones above:

Buttons: the buttons displayed to OEP users allowing them to respond.

Default Button: integer corresponding to the position of the default button as it is displayed in the box. The default button has focus when the box is displayed and registers an itemClick event when the user responds to the prompt by pressing the ENTER key.

Navigation Actions

The actions of the Navigation object display OEP pages. The Navigation object is listed in the action list only, under the Common object.

Action	Description
Get Page Variable	Retrieves the specified page variable.
Navigate To Company	Opens the PowerPage for the specified customer. The company primaryId is the only argument.
Navigate To Company Edit	Opens the company edit page for the specified customer. The company primaryId is the only argument.
Navigate To Email	Opens the email window and displays the specified email message. The email ID argument must be the primaryId of an emailMessage object instance.
Navigate To Help	Opens OEP Help.
Navigate To Home Page	Displays the home page of the OEP user in the data area of the application page.
Navigate To Incident	Opens the incident edit page and displays the specified incident. (Arguments: incident primaryId and category.)
Navigate To Individual	Opens the individual PowerPage for the specified individual. The individual primaryId is the only argument.
Navigate To Individual Edit	Opens the individual edit page for the specified individual. The individual primaryId is the only argument.
Navigate To List Manager	Opens the list manager page.

Action	Description
Navigate To Literature	Opens the literature page for the current customer record.
Navigate To Messenger	Opens the messenger inbox page of the OEP user.
Navigate To Quick Search	Displays the quick search tabs in the data area of the application page.
Navigate To Script Chooser	Opens the Choose a Script page that displays scripts assigned to the specified individual, company, or incident. (Arguments: Owner primary ID, owner secondary ID, owner type, and owner subtype.)
Navigate To Survey	Displays the survey information for the current customer in the data area of the application page.
Navigate To Task	Opens the task edit page and displays the specified task, using the specified primaryId of the task, the incident that owns the task, customer who owns the incident, and the incident category ID of the owning incident.
Navigate To Task Manager	Displays the OEP user's task manager screen in the data area of the application page.
Navigate To Work Ticket	Displays the work ticket page for the specified work ticket (primary Id). If the primaryId of the work ticket is not specified, a blank work ticket page is displayed.

Data Actions

Data actions retrieve XML data from OEAS, retrieve XML data from cached delta merge output, and manipulate strings. The Data object is listed in the action list only, under the Common object.

Action	Description
Call OEAS Method	Makes an OEAS business object method call that returns the returnXml for the method.
Get Delta Merged Document	Returns an XML string that is the result of a merged document for the specified source document.
Get Reference Field Data	Retrieves domain data as an XML string through an OEAS call to the referenceLookup data business object.
Get Xml Attribute Text	Retrieves the value of an XML element attribute.
Get Xml Node Text	Retrieves the text contained within a node of XML.
Replace Text	Searches text for a string and replaces matches with another string.
Set Boolean Return Value	Returns a true or false value that can be stored in a page variable.

Call OEAS Method

This action makes a call to an OEAS business object method that returns the returnXml node for the method. For more information about OEAS methods, see the OEAS Technical Reference.

The arguments listed in the following table are strings.

Arguments	Description
Object Name	The name of the OEAS business object
Method Name	The name of the method on the OEAS business object
Input XML	The well-formed parameters node that is required for the specified method

Get delta Merged Document

This action returns an XML string that is the result of a merged document for the specified source document. The source document is merged with a delta file that specifies UCW configurations.

If there is an error in the merge process and OEP is in a development environment, debug information is returned. If there is an error in a production environment, nothing is returned.

Arguments	Description
URI	The path to the physical or virtual XML file to be merged.
Context Filter	A UCW context string that ensures any page-specific configurations to a common object remains specific to the page and is not common to all users of the source document. For example, the context string "[pageid]=incident;[categoryid]=1;" specifies that the delta merge document configures the incident edit page for support incidents, but does not configure the page for other types of incidents, such as sales and service incidents. Typically, leave this argument empty so that the action picks up the context from the underlying page.
Virtual	Specifies whether the file is virtual or physical. Virtual files do not exist on the file system as a separate file – only in the delta.
Virtual Root XML	A string that specifies the virtual root. Required when the Virtual argument is Yes.

Get Reference Field Data

This action returns domain data as an XML string through an OEAS call to the referenceLookup data business object.

Argument	Description
Reference Field Name	The name of the property that is an instance of the referenceField OEAS business object, for example, incident.priority.
Parent ID	An integer that identifies a parent referenceParameter object instance. This argument is necessary to access hierarchical reference data.
Type	An identifier of the returned data.

Get XML Attribute Text

This action retrieves the value of an XML element attribute.

Argument	Description
Source XML	A string of XML to parse
XPath	An XPath expression that locates the XML element
Attribute Name	The element attribute that contains the value to retrieve

Get XML Node Text

This action retrieves the text contained within a node of XML.

Argument	Description
Source XML	A string of XML to parse
XPath	An XPath expression that locates the XML element

Replace Text

This action searches text for a string and replaces matches with another string.

Argument	Description
Source Text	The text to search
Find Text	The string to match
Replace Text	The string that replaces matched strings

Debug Actions

Debug actions assist in debugging action statements. The Debug object is listed in both the action list and the condition list, under the Common object.

Action	Description
Is Debug Mode?	Returns a boolean that indicates whether debug mode is on. In OEP production mode, this action always returns false.
Is Production Environment?	Returns a boolean that indicates whether if the OEP is a production environment (versus a development environment).
Set Debug Mode	Turns UCW debug mode on or off.
Trace	Creates an entry in the Trace Viewer.

Fire Event

The Fire Event action causes a Dynamic Forms event to fire. This action has the following parameters:

Addressing Mode: The type of addressing mode; for direct mode, the object ID is specified using its ID from the df_framework_definition.xml file; for indirect mode, the UI control ID is specified.

Object ID: The ID of the object.

Event: The type of event, such as load or onClick.

Position: The position (before or after) of the event relative to the existing handler.

First Argument: Optional event argument.

Second Argument: Optional event argument.

Get Page Variable

The get page variable action retrieves the value that a page variable contains. Page variables are available only on the OEP page on which they are created.

To retrieve the value of a variable, provide the variable name.

Important: Type the variable name for the argument: Select **User Defined** from the **Source** drop-down list and then type the name of the variable in the **Value** text box. If you select the variable name instead (using Page Variable from the Source drop-down list to display a list of values in the Value drop-down list), the action looks for a variable with a name identical to the value of the selected variable. For example, if you select msgError from the Value drop-down list (instead of typing "msgError") and the msgError variable contains the value "Error," the action looks for a variable named "Error."

Set Page Variable

The set page variable action specifies the value that a page variable contains. Page variables are available only on the OEP page on which they are created. If you specify a variable that does not exist, the variable is created.

To set the value of a variable, provide the variable name and its value.

Note: Type the variable name for the argument: Select **User Defined** from the **Source** drop-down list and then type the name of the variable in the **Value** text box. If you select the variable name instead (using Page Variable from the Source drop-down list to display a list of values in the Value drop-down list), the action looks for a variable with a name identical to the value of the selected variable. For example, if you select msgError from the Value drop-down list (instead of typing "msgError") and the msgError variable contains the value "Error," the action looks for a variable named "Error" and that variable is created if not found.

Context Data

The following lists specify the context data objects available for each UCW-enabled OEP page. Each object has a Get Value action.

The main application frame and customer product pages do not have context data.

Context data objects for individual PowerPage

- Customer Primary ID
- Customer Secondary ID
- Customer Name
- Customer Type
- Primary Address Line1
- Primary Address Line2
- Primary Address Line3
- Primary Address City
- Primary Address State Code
- Primary Address Post Code
- Primary Address Country Code
- Primary Phone Number

Context data objects for individual edit

- Post Action
- Primary ID
- Secondary ID
- Parent Company
- LBO Return XML

Context data objects for company PowerPage

- Customer Primary ID
- Customer Secondary ID
- Customer Name
- Customer Type
- Primary Address Line1
- Primary Address Line2
- Primary Address Line3
- Primary Address City
- Primary Address State Code

- Primary Address Post Code
- Primary Address Country Code
- Primary Phone Number

Context data objects for company edit

- Post Action
- Primary ID
- Secondary ID
- Primary Contract Primary ID
- LBO Return XML

Context data objects for incident

- Primary ID
- Secondary ID
- Owner Primary ID
- Owner Secondary ID
- Owner Type
- LBO Return XML

Context data objects for task

- Primary ID
- Secondary ID
- Owner Primary ID
- Owner Secondary ID
- Owner Type
- LBO Return XML

Context data objects for product line items

- Primary ID
- Secondary ID
- Owner Primary ID
- Owner Secondary ID

Post action

The Post Action action returns a string that identifies a task performed by the server-side page. This self-post mechanism instructs server-side processes to "post back" information to the client-side OEP page after the server-side request finishes. The post information is accessible through the Post Action action (listed under the Context Data object in the action and condition lists).

Example post action value	Server-side page action
insert	Created a record
update	Updated the record
retrieve	Retrieved the record
delete	Deleted the record
load (or empty context data item)	Initially loaded the page

An example of using the Post Action context data item is to create a conditional action on a load event that displays a message to users telling them that the update is complete.

Control Actions and Conditions

The Control object in the action and condition lists enumerates the UI controls (including toolbox captions) defined for the current page. The condition list includes only the Get Value action for each control. The action list includes a full range of control actions for each control. The following table lists the control actions.

Action	Description
Disable Control	Makes the control display-only.
Enable Control	Enables the user to change or update the value of a control.
Get Code	Retrieves a displayed string that is associated with a value contained by the control. This action is available to Tracking Code controls, for example. Retrieve the control's underlying value using the Get Value action.
Get Control Count	Retrieves the number of controls contained within a collection of controls. Examples of controls that are collection of controls are the Individual_Telephones and Company_ Telephones controls on the PowerPage.
Get Label Text	Retrieves the label of the specified control.
Get Local Date	Retrieves the date held in the control, converted to the client's local date.
Get Local Time	Retrieves the time held in the control, converted to the client's local time.
Get Text	Retrieves the text value that is associated with the underlying integer value contained in the specified control.
Get Value	Retrieves the underlying integer value contained in the specified control.
Give Focus	Sets focus on the specified control. For example, if the control is a text box, the cursor is placed in the text box.

Action	Description
	A control cannot be given the focus if it is hidden, or if the body of the page is hidden. For that reason, if you add the Give Focus action under a page's Load event, make sure that the Give Focus action comes after the Show Body action called from the Onyx layer.
Hide	Hides the specified control.
Hide Details Box	Hides the primary contact details on the incident edit page.
Hide Item	Hides the specified item in a collection-based control, such as a toolbar.
Hide Label	Hides the label of the specified control.
Highlight (Off)	Turns off highlighting of the specified control.
Highlight (On)	Highlights the specified control.
Is Dirty?	Returns a Boolean that specifies whether the value of the control has been changed from its binding data.
Is Recall Set?	Returns a Boolean that specifies whether a reminder is set for the user that is creating or updating an incident or task record.
Is User Reminded?	Returns a Boolean that specifies whether the OEP user assigned to the incident or task is to receive a reminder for record that is currently displayed in a page.
Link *Item*	Displays a window that allows users to associate the record on the page (company, individual, incident, or task) with another record. For instance, link a primary contact (individual record) with the incident record that is displayed on the incident edit page.
Select Item	Selects the specified item in a collection-based control, such as a toolbar.
Set Control Tooltip	Changes the tooltip string of the specified control.
Set Label Text	Changes the label string of the specified control.
Set Local Date	Specifies the date to send reminder messages. The format of the date string is YYYY-MM-DD, and it is the local date rather than GMT.
Set Local Time	Specifies the time to send reminder messages. The format of the time string is hh:mm:ss, and it is the local time rather than GMT.
Set Recall	Adds a reminder to the incident or task currently displayed. The reminder message is sent to the OEP user who is creating or updating the record. When coding this action using Advanced View, the action's argument is typed as a string rather than a Boolean.
Set Reference Data	Sets the reference data for the specified control. This action is available for toolbox controls only. The value pair for the action argument is "referenceData" and the string of data. When the Reselect Value? argument is Yes, the control's current value (if any) is removed and the OEP user reselects a value from the new list of values. If the argument is No, the current value is retained even if it does not exist in the new list of values.

Action	Description
Set Required (On)	Displays an asterisk by the specified control to indicate that a value entry or selection is required for the specified control.
Set Required (Off)	Removes a displayed asterisk by the specified control (indicator for required entry or selection).
Set Tooltip Text	Changes the tooltip string of the specified control.
Set User Reminder	Adds a reminder to the incident or task currently displayed that sends a message to the OEP user that the record is assigned to. When coding this action using Advanced View, the action's argument is typed as a string rather than a Boolean.
Set Value	Changes the value contained in the specified control.
Show	Displays the specified control.
Show Add Dialog	Displays a dialog box that lets users choose to add either an individual or a company customer record.
Show Details Box	Displays the primary contact details on the incident edit page.
Show Item	Displays the specified item in a collection-based control, such as a toolbar.
Show Keywords Dialog	Displays a dialog box that lists the keywords for the displayed incident or task.
Show Label	Display the label of the specified control.
Show Tree	Display a dialog box that lists items in an hierarchical tree and allows users to select an item.
Unlink *Item*	Removes a linked record.

Event Arguments

The Event Argument action let you retrieve the argument of the event that an action is responding to. For example, for the Item Click (Item ID) (Item Value) event on a toolbar control, the Event Arguments (Item ID) action compares the ID to the clicked item on a toolbar, according to the selected operator and value.

Page Variable

You can create Dynamic Forms Designer actions that save results to page variables. The page variables appear in both the action list and the condition list. The actions associated with page variables are Get Value and Set Value (only Get Value is available from the condition list).

Page Actions and Conditions

The following table lists page actions available under the Page object in the action and condition lists. The *item* varies according to the OEP page that is being configured (individual, company, incident, task, product, or product line).

Action	Description
Add New Product Line	Adds a product line to the current product record.
Add New Product Line and Close Window	Adds a product line to the current product record and closes the product line items page.
Attach item to Messenger Message	Opens an OEP messenger composition window and attaches the current record to it.
Change Owner Incident	Displays the quick search dialog box and lets the user find and specify a different owner record.
Clear Controls	Removes data from all controls on the page.
Clear Page Changed	Removes the page dirty indicator for the page.
Delete *item*	Specifies that the record contained on the page should be deleted from the Onyx Enterprise Database (OEDB).
Hide Body	Hides the HTML document body.
Get Current Context Filter	Retrieves the context filter string for the page.
Get Current Page ID	Retrieves the identifier of the OEP page.
Get Current Path to Website Root	Retrieves the relative path to the root of the OEP website.
Get Opener Window Reference	Retrieve the reference (name) of the parent of the current page using the opener property of the window object.
Is Design Mode?	Checks if the page is in design mode.
Link Task Owner	Displays the quick search dialog box and lets the user find and specify a different owner record.
Print Task	Prints the current task.
Save *item*	Specifies that the record contained on the page should be saved to the OEDB.
Save and Close *item*	Saves and closes the page.
Set Alert For *item*	Opens the set alert window, which lets the user create an alert message for the current record.
Set Page Changed	Marks the page as dirty, which causes a not saved warning to appear when a user attempts to navigate away from the page.
Show Body	Displays the HTML document body.

Action	Description
Update Line Item	Saves the line item.

Section Actions

Section actions include hiding and showing sections, which are typically used to group UI controls. The Section object is listed in the action list only.

Sections are identified by their Admin ID properties.

Action	Description
Hide Section	Hide the specified section and the UI controls that it contains. A section is an area within a panel.
Show Section	Display the specified section.

invokeAction function

The invokeAction function is a broker that executes all Dynamic Forms actions on the specified object instance. This is a global function available to all UCW-enabled areas.

```
invokeAction (objectMode, objectId, actionId, [[arguments],])
```

objectMode

[in] A string that specifies the mode in which the object is addressed:

- indirect – Specify the object ID as the UI container ID. To view the ID, enter design mode for the desired UCW-enabled area, right-click the UI control, and then select **Properties**.
- direct – Specify the object type as the UCF type. To view the UCF type, enter design mode for the desired UCW-enabled area, right-click the UI control, and then select **Advanced Properties**. Append the ID to the UCF type to address an existing container.

objectId

[in] A string that identifies the container.

actionId

[in] A string that identifies the action to perform.

arguments

[in] An array of arrays ("ID", "Text" pairs) that contains all optional arguments.

Return values

The return value of invokeAction should always be the object type UcfActionReturn. The object standardizes action execution, return data, and error reporting for Dynamic Forms Designer.

The following snippet shows the code that Dynamic Forms Designer generates for a Get Event Argument action that saves the result to a page variable. Note that the value property of the UcfActionReturn object stores the result of the action, which is the event argument. Also, Stop on Execute is specified for the action (using the standard view of Dynamic Forms Designer), so the stop property is set to true.

```
// ACTION:  ActionID=getArgument1,  ObjectID="ucf.eventArgument"
// Summary: Get Event Argument
oUcfActionReturn = invokeAction("direct", "ucf.eventArgument",
"getArgument1",
            new Array("eventArgument1", psEventArgument1),
             new Array("eventArgument2", psEventArgument2));
oUcfActionReturn.stop = true;
// save result in page variable: myVariable
invokeAction("direct", "ucf.variable:MyVariable", "setValue",
      new Array("value", oUcfActionReturn.value));

// terminate event if stop set, and return action object
if (oUcfActionReturn.stop) {
return oUcfActionReturn;
}
```

Remarks

Action statements generated using the standard view of Dynamic Forms Designer alter and create OEP page behavior wholly through the invokeAction broker. When using Advanced View, use invokeAction calls wherever possible and avoid direct calls to the underlying page. Script not passing through the broker may require manual steps to re-implement the configuration when upgrading to new releases of OEP.

If an instance of the specified object type is not found, one is created and the specified action executed. In the preceding code snippet that demonstrates return values, a variable object type instance is created for MyVariable and the contents of oUcfActionReturn are stored in it.

3

Customizing OEP

Overview

If you want to modify OEP areas that cannot be modified using UCW, you will need to do so by writing custom code. You can write custom code for any area that is not enabled by UCW and, of course, for any new page that you add to OEP.

Warning: Use UCW to modify UCW-enabled OEP areas. Do not write custom code directly to these areas, or unpredictable results may occur.

Customization reference materials

This book provides information about how to customize OEP, guidelines for performing and managing these customizations, and a number of typical customization examples. Note that many tasks that previously could be done only by writing custom code can now be accomplished using UCW.

The OEP application details provides information on the key programming features of the OEP design, and how these features operate and can be used in customizations.

The examples in this customization guide illustrate various customization techniques, recommended practices, and programming feature demonstrations.

The Programmer's Reference contains information on OEP functions, classes, and COM components that can be used in customizations.

Programming tools and skill requirements

The skills and tools required depend on the type of customization you will be performing. The following table lists the general skill and knowledge requirements related to the OEP customization examples in this guide.

Tool	Required for
Onyx Enterprise Studio Administration Tools	Configuring Onyx user accounts Configuring certain aspects of OEP, such as by using Table Administration to add list manager views
Microsoft Visual Studio	Script and style sheet editing, debugging
VBScript	Script modifications
JavaScript	Script modifications

Files Location

The files required to run these examples are located in the following directories on the OEP installation CD:

Files	Location
OEP website	Support\WebSite
Customization Guide ASP examples	Documentation\Technical Reference\Examples

Note: The customization examples must be saved to a subdirectory named Examples within the directory that contains the OEP website. For installation instructions, see the Readme.txt file in the Examples directory.

OEP Application Details

This section describes the design and implementation of key features of the OEP application. The information in this section provides programmers with a foundation of understanding upon which to base customization decisions.

Feature	Description
Authentication and security	Discusses the authentication and security control within the Onyx Security and Licensing Framework (OSLF)
Session management	Describes how sessions are created and maintained
Cached data	Describes the functions that provide for data persistence within OEP and user sessions
Masked editing	Describes how the masked editing components work and how they can be used in customizations
Using OTMHelper classes	Describes VBScript classes that provide assistance in interacting with the OEAS Onyx Transaction Manager (OTM)

Authentication and Security

OEAS uses the Onyx Security and Licensing Framework (OSLF) for user authentication and permissions. OSLF components support two types of authentication: Windows (NT) authentication and Onyx (database) authentication. OEP users can log in using one or both of the supported authentication types. Permissions to access individual functions within OEP are configured using the *role and resource model*. For more information about configuring these permissions, refer to both the OEAS Technical Reference and the OES Security Administration online Help.

OEP encapsulates this authentication processing in the *initializePageSecurity* and *initializePageSecurityInline* functions.

The initializePageSecurity function is called at the top of every ASP page that requires access control. If the call fails, the user is automatically redirected to a login page where they must establish an authorized session. If the call succeeds an OEPSession object is established, and the server proceeds with the ASP page processing.

The initializePageSecurityInline function operates similarly to the initializePageSecurity function with the exception that it returns a boolean indicator rather than automatically redirecting to the login page. This allows the application to manage authentication failures within the application logic.

All features within OEP are secured through user interface resources that are editable using OES Security Administration. By default, the standard OEP roles have access to all available resources.

Once the session has been validated, access rights for a given user can be checked by calling the *HasPermission* method of the user's OEPSession object, passing the name of the desired resource as a parameter. The method determines the access rights by passing the request to the OSLF session manager, which returns a boolean indicator for the requested resource access. By breaking the available actions of a page into separate permissions (for example, insert, update, delete), you can configure a single interface to provide only the links that the user is allowed to access.

OEP UI resources are named according to feature and function. All begin with the prefix UI:OEP. For example, the resource to add company records is **UI:OEP.company.insert**. Information about the default resources in OEP is available in the OEP Technical Guide.

Session Management

An OEP session is created when a user is granted access to OEP by the Onyx Security and Licensing Framework (OSLF) components. The first time a user encounters the *intializePageSecurity* or *initializePageSecurityInline* functions in the OEP ASP pages, OSLF authentication processing is performed.

If the authentication processing is successful, an *OEPSession* object is created for the user that contains the user's credentials and other session-related information. The information contained in the OEPSession object is used throughout the OEP application to determine if the user has access rights to restricted resources.

The OEPSession class manages data and provides features that extend the standard ASP Session object. Use OEPSessions for all client-connection management tasks.

Cached Data

OEP's cached data architecture provides a means for storing commonly used OEP application data. The caches improve performance and scalability by eliminating redundant requests to the database.

OEP uses four distinct caches, each with a unique purpose:

Cache name	Description
Application	Contains data that is not expected to change while the application is active.
Context	Contains data that is likely to change during the application lifetime, or needs to be accessible from the client side.
User preferences	Contains a user's application preference information.
XCacheXml	Client-side XML based cache. Instantiated as "goGlobalCache" on powerpage/application_main.asp. Used to cache XML data on the client for the duration of the user session. Data cannot be accessed directly on the server.

Each of these caches can be accessed through a different set of OEP API functions. Refer to the cached data section of the Programmer's Reference for information about using the cached data functions for moving data in and out of the caches.

Note: Using the OEP diagnostic tools, you can view the contents of the application and context caches (and in some cases, can refresh the caches).

Application Cache

The application cache contains data that is not expected to change while the application is active. The majority of the data located in this cache is stored there during system initialization. Additional elements are added as users log in to the system and certain portions of the application are used.

Each OEP feature adds data to the cache as the system initializes. The features then refer back to the cache as they generate pages for client browsers. The data stored in the cache falls into one of the following categories:

- System configuration and initialization data. This includes startup time and duration, database connection information, system version data, and server information.
- Default user preferences. System-wide defaults are stored here. A user who does not set a value for a preference inherits a setting from these defaults.
- OEAS system parameters. These are OEP-specific parameters that are configured through the System Parameter administration tool. Refer to the OEAS documentation for information about these parameters.
- User-defined field (UDF) metadata. Information about the available UDFs is stored here. Each field has two entries. These are described below.
- Domain data. Each feature stores lists of available domain data for the list boxes and tree controls that appear in the user interface. This includes the priority, status, and source choices for incidents and work tickets, as well as region and country codes, postal and telephone masks, user lists, product information, and survey choices.
- List Manager views. Information about the selectable fields and SQL for List Manager views is stored here.
- User-specific data. As each user logs in to OEP, information about their system permissions and non-default preferences is added as needed.

Data Formats

Data cached on the server is stored either as a string or as an array. Code that retrieves data from the cache should use the VBScript IsArray function before attempting to read the data. Many of the arrays are two-dimensional, but all can be checked for size using the UBound and LBound functions. The following code sample can be used to determine the number of dimensions within an array object.

```
lCols = empty
lRows = ubound(vValue,1)
```

```
on error resume next
lCols = ubound(vValue,2)
err.clear
on error goto 0

if IsEmpty(lCols) then
' vValue is a 1 dimensional array
else
' vValue is a 2 dimensional array
end if
```

Although the default configuration of OEP does not do so, the Application cache can also store objects created in ASP scripts.

Data in string formats may contain well-formed XML documents or document fragments.

UDF pairs

Each user-defined field is represented by two separate arrays in the application cache.

- The first entry ends with the string .DETAILS, and identifies the type of user interface object the field should use (combo box, text box, free text etc.), and configuration data specific to that type.
- The second entry ends with the string .DATA and contains a list of domain values to use for the field. If the field's type does not support domain data, the array is empty.

Use the function vbPrintUDF in the file YourOEPwebsite/common/include/RenderUDF.asp to automatically add UDF fields to user interface elements.

Reviewing the cache contents

Use the OEP diagnostics tools to view the contents of the application cache. From there you can review and refresh the data cached on the OEP server.

Context Cache

The context cache provides a storage location for information that is passed back and forth between the client browser and the OEP server. Typical uses for this cache include session information, locale identifiers, and the unique ID of the active customer record. Data within this cache is readable from both the client and server and can be updated at any time during the lifespan of the OEAS session.

OEP stores the context cache data within browser cookies. Because of this, the total amount of data that can be stored in the cache is limited to four kilobytes per client. OEP stores a little less than one kilobyte in this cache in its default configuration.

To save space, much of the data within the context cache is identified using abbreviated keys. Items in the cache can be a single data item paired with a key, or a group of associated items assembled

under a parent key. There are functions for saving and retrieving the data in either format. Information about the functions and the keys is available in the context cache section of the Programmer's Reference.

User Preference Cache

The user preference cache has only one function, *UserPrefCacheRead*, that retrieves information about the user's preferences. There is no function to store user preference information. This data is set by the OEP application.

Data for user preferences can be retrieved using keys that identify the preference. A complete list of keys is available within the file YourOEPwebsite/common/include/otm_cache.asp. Some sample keys appear in the table below.

Key	Description
CONFIRMCMPSAVE	Determines whether or not to display a confirmation message after a successful save to a company record
EXPGRPSWITHSELUSERS	Determines whether or not to expand the groups with selected users within the user control
SUPPORTINCASSIGNEDTO	Contains the default value for the Assigned To field of a support incident

The case of key names is not important and is ignored by the *UserPrefCacheRead* function.

Masked Editing

Masked editing is a means by which you can:

- Restrict users from entering certain characters into certain fields on the user interface.
- Format data that is returned from the database, so that it displays as desired in the user interface.

There are two aspects to masked editing: applying mask rules and formatting the results.

Applying masks rules

The characters entered by the user are compared to a rule loaded into the control. A rule contains information that indicates what type of character is allowed at any given position. For example, a rule that contains "######" specifies a requirement of five digits. No letters or punctuation are allowed, nor may any characters be left blank.

Formatting returned data

A masking rule can also define the format of text with the inclusion of literal characters. For example, if a rule contains "###-####", the dash ("-") character is a literal that cannot be edited by the user. The rule requires that the user enter seven digits. The resulting output contains those digits with the literal dash character between the third and the fourth digits.

Onyx masked editing objects

Two components are available for performing masked editing of user-entered text:

Component	Description
Masked edit control	An ActiveX control for use in interactive client-side applications.
Masked data component	A COM object for use in either server-side or client-side applications.

The characters used to create masks are explained in the Masking characters topic.

Masking Characters

The characters for a mask are interpreted according to the mode of the masking component. There are two modes, Standard and Postal. The mode is set through the MaskRules property.

This information is intended only for use for masking in OEP. OEAS also supports masking, but it does not support the same mask rules as OEP. More specifically, OEAS does not support postal masking except for a very limited set of characters. See the *OEAS Developer's Reference* for information on which masking characters are supported when masks are applied during business object method calls.

Important: Leading spaces in a data field are often trimmed by the application server when the contents of the field are written to the database. If you create a mask that allows spaces as the first user-enterable items in a masked data field, they will not be saved when the data is saved, causing unexpected behavior when the data is later retrieved and displayed to the user.

The following table shows the supported characters for the **Standard** mode of the Onyx masking components.

Mask character	Description
9	Digit (0-9) or decimal point.
A, a	Any uppercase or lowercase letter (A-Z or a-z).
D, d	Digit (0-9) or space.
H, h	Any hexadecimal character (0-9, A-F, or a-f).
L, l	Any letter (A-Z or a-z). If an uppercase letter is input, it is converted to lowercase.
N, n	Any alphanumeric character (0-9, A-Z, or a-z).
U, u	Any letter (A-Z or a-z). If a lowercase letter is input, it is converted to uppercase.
X, x	Any character, including Unicode characters on systems that support Unicode. Use X for all far-east Unicode characters.
/	Converts a mask character to a literal. To display a forward-slash character as a literal, use two consecutive forward-slash characters. For example "D//D" displays a digit entry position, followed by a single forward slash, followed by another digit entry position.

For the **Postal** mode, the mask characters are interpreted as follows:

Mask character	Description
#	Digit (0-9).
&	Character placeholder. Valid values for this placeholder are the ANSI characters in the following ranges: 32-126 and 128-255.
,	Digit group separator.
.	Decimal separator.
/	Slash. Date separator.
:	Time separator.
<	Convert all following characters to lowercase. The < character does not appear visually in the control, but causes all characters following it to be converted to lowercase as they are entered.
>	Convert all following characters to uppercase. The > character does not appear visually in the control, but causes all characters following it to be converted to uppercase as they are entered.
9	Digit (0-9), or space.
A, a	Any alphanumeric character (0-9, A-Z, or a-z).
C, c	Character or space placeholder. This operates exactly like the & placeholder, and ensures compatibility with Microsoft Access.
?	Any uppercase or lowercase letter (A-Z or a-z).
\	Backslash. Converts a mask character to a literal. To display a backslash character as a literal, use two consecutive backslash characters. For example "#\\#" displays a digit entry position, followed by a single backslash, followed by another digit entry position.

All other characters or symbols are treated as literals and appear in the masked text as shown to the user. Use punctuation marks as separators for date, time, and numbers, as required by your international settings.

Warning: Do not use the literal identifier (forward slash for Standard mode and backslash for Postal mode) in front of another literal (non-mask) character. Unexpected results may occur. The literal identifier should only be paired with one of the masking characters that appear in the two tables in this topic.

Masked Edit Control

The masked edit control that can be employed in client-side browser scripting applications. It is a visual control that behaves as a standard text input control, with the exception that the mask rules determine its behavior. The masking and formatting rules are applied as the user enters characters in the control, and the results are displayed interactively. When the user attempts to enter a character that is not allowed by the masking rules, the control refuses to accept the character and fires a ValidationError event.

The mask rules can contain literal characters. These characters appear in the edit field, but the user cannot change them. The insertion point automatically skips these characters as the user enters text.

Using OTMHelper Classes

The OTMHelper classes abstract interaction with the Onyx Transaction Manager (OTM). These classes provide methods for establishing a logical connection with the OTM, packaging requests for business object operations, and processing the resulting data. The OEP architecture is designed to work with record-oriented data. The OTMHelper classes serve as a means for dealing with business objects, which are represented in XML data structures, in a record-oriented manner.

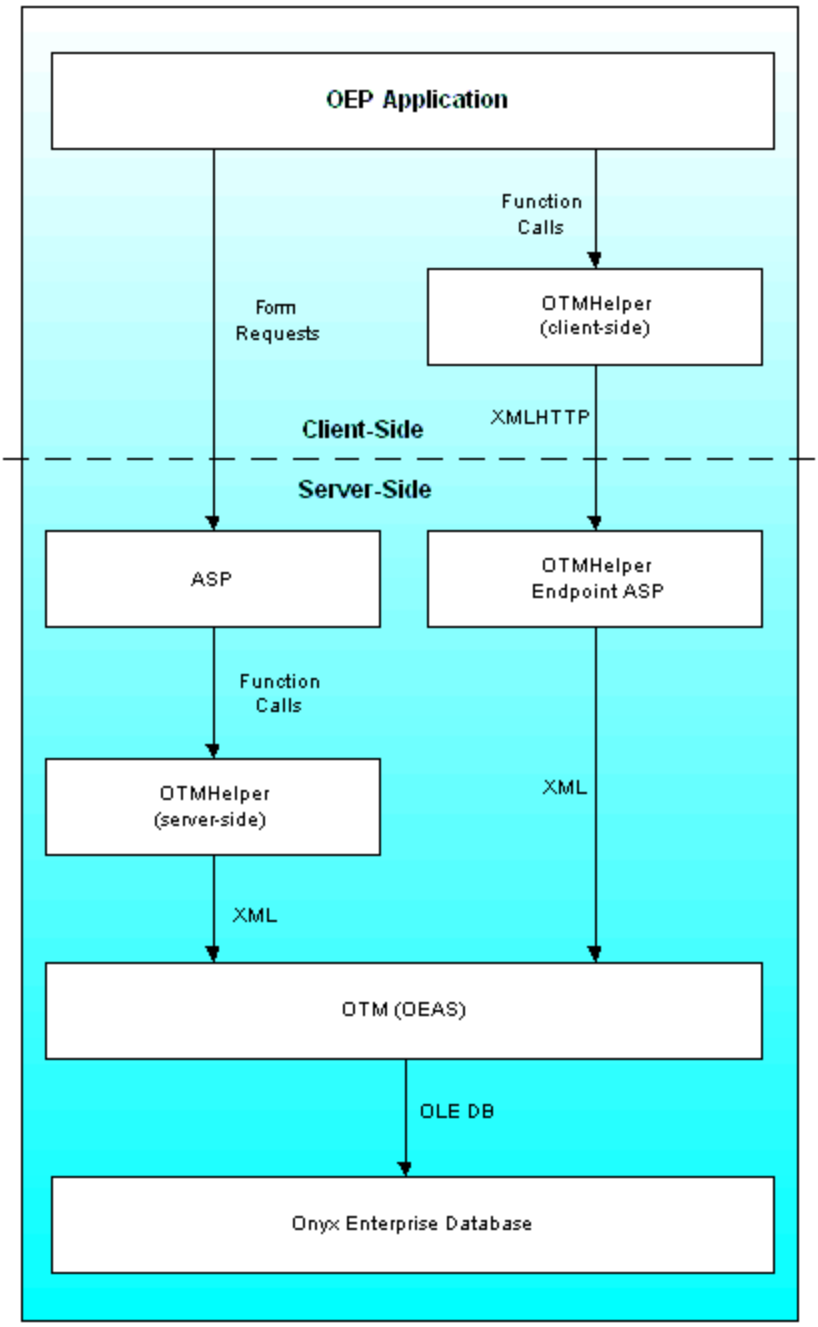
> **Important:** This section assumes you are familiar with the underlying principles of interacting with the OTM. You should be familiar with the OEAS Developer's Reference and have an understanding of how business objects operate.

There are two versions of the OTMHelper classes, server-side and client-side. The two versions are identical, apart from the two methods *bExecuteLocal* and *bExecuteRemote*. For this reason, code that uses the OTMHelper classes can be implemented on either the client or server side, and moved back and forth easily.

ASP pages use the server-side OTMHelper classes. These scripts create an XML document on the server side, pass it to an instance of the Onyx Transaction Manager object, and process the resulting XML data before sending HTML to the client.

The client-side OTMHelper classes are used in browser scripting. They create an XML document and pass it to a special server-side ASP page via XMLHTTP, which in turn passes it to the Onyx Transaction Manager object. The resulting XML data is returned to the browser for formatting and/or processing.

Server-side/client-side architecture comparison diagram



Using the classes

The OTMHelper classes are very versatile. One typical use is demonstrated in the following sequence:

1. An OTMConnection object is created and initialized to establish communications with the OTM.
2. An OTMLBOCall object is created through the OTMConnection object. The OTMLBOCall object is then initialized by specifying the business object and method that this object will represent.

3. Input data for the method call is created using the OTMLBOCall AddParam and bAddObjParam methods.
4. The OTMLBOCall object's ExecuteLocal or ExecuteRemote method is called, depending on whether the request is running on the client or the server.
5. If the operation results in data retrieval from OEAS, an OTMRowset object contains the XML information in a record-oriented representation.

OEAS Object Parameter Attributes

The following tables summarize the OTM action and content attributes used on OEAS business objects. See the OEAS Technical Reference for detailed information about the use of these attributes.

action attribute

The action attribute is used with business object saveCollection methods. The value of the attribute indicates the database action for the business object.

Name	Description
insert	Saves the business object data in the Onyx Enterprise Database (OEDB).
update	Updates the existing business object data in the OEDB.
delete	Deletes a business object from the OEDB.

content attribute

The content attribute indicates what properties are present for a business object.

Name	Description
all	Indicates that all the properties defined for the business object in the OED are present. This does not mean that all properties contain data, just that they exist.
any	Indicates that a subset of properties defined for the business object in the OED are present and any of them may be of use for the requested action.
keysOnly	Indicates that all the properties defined as keys for the business object in the OED are present.
partial	Indicates that a specific subset of properties of the business object in the OED are present.

Although 'partial' and 'any' may appear to represent the same thing, they are different. The 'partial' attribute indicates a business object that contains a specific and predetermined set of properties necessary for a business object method. The 'any' attribute indicates a set of properties that can vary between calls to the same business object method. See the OEAS Technical Reference for a more in depth discussion of the content attribute and its possible values.

OTMHelper Usage Guidelines

The following guidelines are provided to enable you to use the OTMHelper classes most effectively.

Path to rowSet objects

Use the following syntax to obtain a rowset:

```
oCall.bGetRowset("rowSet", oRst, sMethodStatus)
```

Although it is possible to obtain a rowset using the following syntax, it is not recommended because you are not accessing the full functionality of the OTMRowset object:

```
oCall.bGetRowset("rowSet/rows", oRst, sMethodStatus)
```

Using the sort methods

The bSort and bAdvancedSort methods of the cOTMRowset class have been designed to be as efficient as possible, but their use on the OEP server should be kept to a minimum, as they can put significant resource demands on the server when executing. Check to see if the rowset of the business object is already sorted and use one of the methods only if needed.

Both methods support Unicode data sets and honor the locale sort order of the client user, even when running on the server.

Use client-side execution

When used appropriately, client-side execution of business object method calls can reduce server load and improve overall application performance. As mentioned above, rowset sorting can put significant resource demands on the OEP server. By using the client-side versions of the sort methods, you can offload the sorting burden onto the client for improved performance.

The OEP Customization Guide contains examples that demonstrate how to perform an equivalent operation using both the server-side and client-side versions of the OTMHelper classes.

Leverage the existing XMLDOMDocument object

Because the cOTMLBOCall class contains an already-parsed XMLDOMDocument object, it is not necessary to create one from the raw XML property. If the cOTMRowset classes do not provide necessary functionality, access the document data directly using the cOTMLBOCall methods bGetXMLDOC and bGetParamNode.

OTMHelper Examples

This section contains simple examples that demonstrate how to perform common operations using the OTMHelper classes. The following examples are provided:

Example	Description
Simple retrieve	How to retrieve a simple data list
Multiple object retrieve	How to retrieve a business object that has collections of child objects
Object insert	How to insert a business object

Example	Description
Object update	How to update a business object
Object delete	How to delete a business object
Collection retrieve	How to retrieve a collection
Collection insert	How to insert an item into a collection
Collection update	How to update an item in a collection
Collection delete	How to delete an item in a collection
Combined collection operations	How to perform mixed operations on multiple items in a collection

Simple Retrieve Example

The following example demonstrates how to perform a simple retrieval using the OTMHelper classes. This example performs a keyword list retrieval, which is placed into an OTMRowset object for application use.

```
Sub TestCallKeyWord(sApp, sSessID)
Dim oCon, oCall, oRst
Dim sMethodStatus, sStatusType
Set oCon = New cOTMConnection
If oCon Is Nothing Then
' error out
End If
If Not oCon.bInitialize(sApp, sSessID, sMethodStatus) Then
' error out
End If
If Not oCon.bCreateLBOCall(oCall, sMethodStatus) Then
' error out
End If
' Intialize call object
If Not oCall.bInitialize("keyWord", "retrieveList", "", "",
sMethodStatus) Then
' error out
End If
' Add input parameters
oCall.AddParam "incidentCategory", 1
```

```
' Make the call
If Not oCall.bExecuteLocal(sStatusType, sMethodStatus) Then
' error out
End If
' Get rowset from output params
If Not oCall.bGetRowset("rowSet", oRst, sMethodStatus) Then
' error out
End If
If Not oRst.BOF Then
' Returned data is in the oRst OTMRowset object
Else
' No rows in rowset
End If
End Sub
```

Multiple Object Retrieve Example

The following example demonstrates how to perform a more complex retrieval using the OTMHelper classes. This example performs a retrieval of a company business object and its address and phones collections. The results of this retrieval are placed into separate OTMRowset objects for company, address, and phones information.

```
Sub TestCallCompany(sAppName, sSessID)
Dim oCon, oCall
Dim sMethodStatus, sStatusType
Dim oObjCompany, oObjAddresses, oObjPhones
Dim oRstCompany, oRstAddresses, oRstPhones
Set oCon = New cOTMConnection
If oCon Is Nothing Then
' Error out
End If
If Not oCon.bInitialize(sAppName, sSessID, sMethodStatus) Then
' Error out
End If
' Get OTM call object
If Not oCon.bCreateLBOCall(oCall, sMethodStatus) Then
```

```
' Error out

End If

' Initialize call object

If Not oCall.bInitialize("company", "retrieve", "", "",
sMethodStatus) Then

' Error out

End If

' Add input parameters

If Not oCall.bAddObjParam("company", oObjCompany, "keysOnly",
False, Null, sMethodStatus) Then

' Error out

End If

oObjCompany.AddProp "primaryId", "4B034AC2-0FE6-425F-AEE4-
4475A9D80054"

If Not oObjCompany.bAddObjProp("phones", oObjPhones, "partial",
True, "phone", sMethodStatus) Then

' Error out

End If

oObjPhones.AddColItem "", "partial"

oObjPhones.AddProp "ownerId", "4B034AC2-0FE6-425F-AEE4-
4475A9D80054"

oObjPhones.AddProp "ownerType", "1"

If Not oObjCompany.bAddObjProp("addresses", oObjAddresses,
"partial", True, "address", sMethodStatus) Then

' Error out

End If

oObjAddresses.AddColItem "", "partial"

oObjAddresses.AddProp "ownerId", "4B034AC2-0FE6-425F-AEE4-
4475A9D80054"

oObjAddresses.AddProp "ownerType", "1"

If Not oCall.bExecuteLocal(sStatusType, sMethodStatus) Then

' Error out

End If

' Get the Company rowset from output parameters
```

```
If Not oCall.bGetRowset("company", oRstCompany, sMethodStatus)
Then
' Error out
End If
' Get the Company's Addresses collection and put it in a rowset
If Not oRstCompany.bGetRowset("addresses", oRstAddresses,
sMethodStatus) Then
'No reason to error out - there's nothing we can do about it.
'Just make a note that there are not addresses to display.
Else
' oRstAddresses now holds the rowset for the company's addresses
list
End If
' Get the Company's Phones collection and put it in a rowset
If Not oRstCompany.bGetRowset("phones", oRstPhones,
sMethodStatus) Then
'No reason to error out - there's nothing we can do about it.
'Just make a note that there are no phones to display.
Else
' oRstPhones now holds the rowset for the company's phones list
End If
End Sub
```

Object Insert Example

The following example demonstrates how to insert a new reminder object.

Note: When inserting a business object, you must provide all of the object's properties even if you don't have data for all of them.

```
Sub TestInsertReminder(sAppName, sSessionId)
Dim oCon, oCall, oObj, oRst
Dim sMethodStatus, sStatusType
Dim sId ' This will contain the primaryId for the new reminder.
Set oCon = New cOTMConnection
If oCon Is Nothing Then
```

```
' Error out
End If
If Not oCon.bInitialize(sAppName, sSessionId, sMethodStatus)
Then
' Error out
End If
' Get OTM call object
If Not oCon.bCreateLBOCall(oCall, sMethodStatus) Then
' Error out
End If
If Not oCall.bInitialize("reminder", "insert", "", "",
sMethodStatus) Then
' Error out
End If
' The input parameters consist of a single reminder object
If Not oCall.bAddObjParam("reminder", oObj, "all", False, "",
sMethodStatus) Then
' Error out
End If
' Very tedious, but necessary, we must provide all of the
properties.
' Many of those left empty will be populated by OEAS when the
data is saved.
oObj.AddProp "primaryId", ""
oObj.AddProp "secondaryId", ""
oObj.AddProp "ownerId", "32D55CA1-F091-45D9-B544-3F46CBFDD058"
oObj.AddProp "ownerType", "6"
oObj.AddProp "userId", "sa"
oObj.AddProp "message", "Message text"
oObj.AddProp "dueBy", "2002-10-31 12:15:34"
oObj.AddProp "recurType", 0
oObj.AddProp "recurInterval", 0
oObj.AddProp "alarmActive", 0
oObj.AddProp "insertBy", Null
```

```
oObj.AddProp "insertDate", Null
If Not oCall.bExecuteLocal(sStatusType, sMethodStatus) Then
' Error out
End If
' The output is also a reminder object, we can treat it as a
rowSet with a single row
If Not oCall.bGetRowset("reminder", oRst, sMethodStatus) Then
' Error out
Else
sId = oRst("primaryId")
End If
End Sub
```

Object Update Example

The following example demonstrates how to update the message property of a reminder object. The XML from a function call that fetched the original object data to be updated is stored in the variable sXML. By using the bInitializeMerge method instead of the bInitialize method, you can use existing XML data as a basis for the update data.

As you can see from this code, updating a single property on an object, while retaining custom properties that your code need not be aware of, is very straightforward when using the OTMHelper classes.

```
Sub TestUpdateReminder(sAppName, sSessionId)
Dim sMethodStatus, sStatusType
Dim sXML, oRst
Dim oCon, oCall
Set oCon = New cOTMConnection
If oCon Is Nothing Then
' Error out
End If
If Not oCon.bInitialize(sAppName, sSessionId, sMethodStatus)
Then
' Error out
End If
' Get OTM call object
If Not oCon.bCreateLBOCall(oCall, sMethodStatus) Then
```

```
' Error out
End If
'....
' sXML is loaded by a retrieve call for a given reminder.
' See the Simple retrieve example for how this is done.
' sXML = othersource()
'....
If Not oCall.bInitializeMerge(sXML, "reminder", "update", "",
"", sMethodStatus) then
' Error out
End If
' We only have to set the one property, after using
bInitializeMerge for the unchanging ones.
If Not oCall.bGetRowset("reminder", oRst, sMethodStatus) Then
' Error out
Else
oRst.Fields("message") = "New message text!"
End If
If Not oCall.bExecuteLocal(sStatusType, sMethodStatus) Then
' Error out
End If
End Sub
```

Object Delete Example

The following example demonstrates how to delete a reminder object. The delete method of the reminder business object requires a partial object that contains the primaryId, ownerId, and ownerType properties.

```
Sub TestDeleteReminder(sAppName, sSessionId)
Dim oCon, oCall, oRst, oObj
Dim sMethodStatus, sStatusType
Set oCon = New cOTMConnection
If oCon Is Nothing Then
' Error out
End If
```

```
If Not oCon.bInitialize(sAppName, sSessionId, sMethodStatus)
Then
' Error out
End If
If Not oCon.bCreateLBOCall(oCall, sMethodStatus) Then
' Error out
End If
If Not oCall.bInitialize("reminder", "delete", "", "",
sMethodStatus) Then
' Error out
End If
If Not oCall.bAddObjParam("reminder", oObj, "keysOnly", False,
Null, sMethodStatus) Then
' Error out
End If
oObj.AddProp "primaryId", B70C3A91-E8DD-4621-B47A-6C4E451B70EF
If Not oCall.bExecuteLocal(sStatusType, sMethodStatus) Then
' Error out
End If
End Sub
```

Collection Retrieve Example

The following example demonstrates how to retrieve a phones collection. Collection retrievals require the creation of a business object collection that contains a single object with properties that the retrieved items will match.

```
Sub TestCollectionRetrieve(sAppName, sSessionId)
Dim oCon, oCall, oRst, oObj
Dim sMethodStatus, sStatusType
Set oCon = New cOTMConnection
If oCon Is Nothing Then
' Error out
End If
If Not oCon.bInitialize(sAppName, sSessionId, sMethodStatus)
Then
```

```
' Error out
End If
If Not oCon.bCreateLBOCall(oCall, sMethodStatus) Then
'Error out
End If
If Not oCall.bInitialize("phone", "retrieveCollection", "", "",
sMethodStatus) Then
'Error out
End If
'To retrieve a collection of phones, input a single phone object

'with the ownerId and ownerType properties set.
If Not oCall.bAddObjParam("phones", oObj, "", True, "phone",
sMethodStatus) Then
'Error out
End If
oObj.AddColItem "retrieve", "any"
oObj.AddProp "ownerId", 1956C3F4-D725-4B7A-9858-516FB68AA349
oObj.AddProp "ownerType", 2
If Not oCall.bExecuteLocal(sStatusType, sMethodStatus) Then
'Error out
End If
If Not oCall.bGetRowset("phones", oRst, sMethodStatus) Then
'Error out
Else
'oRst now holds the rowset for the phones list
End If
End Sub
```

Collection Insert Example

The following example assumes that you have already performed a retrieveCollection, and have retained the XML output generated by the call in the variable sXML. A new phone item is inserted into the phones collection, and the primaryId for the newly created phone item is returned in the external variable sPhoneId.

```
Sub TestCollectionInsert(sAppName, sSessionId)
Dim sMethodStatus, sStatusType
Dim sXML, oRst, oCon, oCall
Set oCon = New cOTMConnection
If oCon Is Nothing Then
'Error out
End If
If Not oCon.bInitialize(sAppName, sSessionId, sMethodStatus)
Then
'Error out
End If
If Not oCon.bCreateLBOCall(oCall, sMethodStatus) Then
'Error out
End If
' ....
'sXML is set by a retrieve call for an similar collection.
'See the Collection retrieve example for how this is done.
'sXML = othersource()
'....
'Use the XML from the retrieve to build the input data for the
save.
If Not oCall.bInitializeMerge(sXML, "phone", "saveCollection",
"", "", sMethodStatus) Then
'Error out
End If
If Not oCall.bGetRowset("phones", oRst, sMethodStatus) Then
'Error out
End If
'To save OTM some work, Strip out the previously retrieved phone
objects from the collection.
'This does not delete them from the database, only from the
working rowset object.
While Not oRst.EOF
If Not oRst.bRemoveRow(sMethodStatus) Then
```

```
' Error out
End If
Wend
'Add the new row for the new phone
If Not oRst.bAddRow("insert", "all", sMethodStatus) Then
'Error out
End If
'Set the new phone object properties
With oRst
.Fields("primaryId") = ""
.Fields("primary") = 0
.Fields("ownerId") = E5CDF90B-2C8E-4DED-9E68-8183622CE8EE
.Fields("ownerType") = 2
.Fields("phoneTypeId") = 115
.Fields("phoneNumber") = "2065551212"
.Fields("readOnlyAccess") = 0
.Fields("privateAccess") = 0
.Fields("insertBy") = ""
.Fields("insertDate") = ""
.Fields("updateBy") = ""
.Fields("updateDate") = ""
.Fields("locked") = ""
.Fields("onyxTimestamp") = ""
End With
If Not oCall.bExecuteLocal(sStatusType, sMethodStatus) Then
' Error out
End If
'Retrieve the output rowset information to get the new item's
primaryId
If Not oCall.bGetRowset("phones", oRst, sMethodStatus) Then
'Error out
Else
```

```
'Get the ID for the new phone object
sPhoneId = oRst("primaryId")
End If
End Sub
```

Collection Update Example

The following example assumes that you have already performed a retrieveCollection, and have retained the XML output generated by the call in the variable sXML. An existing phone item is modified and the phones collection is updated.

```
Sub TestCollectionUpdate(sAppName, sSessionId)
Dim oCon, oCall, oRst, oObj
Dim sMethodStatus, sStatusType
Dim sId 'This is the ID of the phone record to update.
sId = "1"
Set oCon = New cOTMConnection
If oCon Is Nothing Then
'Error out
End If
If Not oCon.bInitialize(sAppName, sSessionId, sMethodStatus)
Then
'Error out
End If
If Not oCon.bCreateLBOCall(oCall, sMethodStatus) Then
'Error out
End If
'....
'sXML is set by a retrieve call for an similar collection.
'See the Collection retrieve example for how this is done
'sXML = othersource()
>'....
'Use the XML from the retrieve to build the input data for the
save.
If Not oCall.bInitializeMerge(sXML, "phone", "saveCollection",
"", "", sMethodStatus) Then
```

```
'Error out
End If
If Not oCall.bGetRowset("phones", oRst, sMethodStatus) Then
'Error out
End If
While Not oRst.EOF
'Step through the rowset looking for the phone we want to update
If oRst.Fields("primaryId") = sId Then
'Found it, update and move on
oRst.Fields("phoneNumber") = "4255551212"
'Tell OTM to update this object
If Not oRst.bSetRowProps("update", "all", sMethodStatus) Then
'Error out
End If
oRst.MoveNext
Else
'To save OTM some work, strip out all the previously retrieved
phones
'from the collection, except for the one to be updated.
'This does not delete them from the database, only from the
working rowset object.
If Not oRst.bRemoveRow(sMethodStatus) Then
'Error out
End If
End If
Wend
If Not oCall.bExecuteLocal(sStatusType, sMethodStatus) Then
'Error out
End If
End Sub
```

Collection Delete Example

The following example assumes that you have already performed a retrieveCollection, and have retained the XML output generated by the call in the variable sXML. An existing phone item with a

primaryId given in the variable sId is deleted from the phones collection.

```
Sub TestCollectionDelete(sAppName, sSessionId)
Dim oCon, oCall, oRst, oObj
Dim sMethodStatus, sStatusType
Dim sId 'This is the ID of the phone record to delete.
sId = "1"
Set oCon = New cOTMConnection
If oCon Is Nothing Then
'Error out
End If
If Not oCon.bInitialize(sAppName, sSessionId, sMethodStatus)
Then
'Error out
End If
If Not oCon.bCreateLBOCall(oCall, sMethodStatus) Then
'Error out
End If
'....
'sXML is set by a retrieve call for an similar collection.
'See the Collection retrieve example for how this is done.
'sXML = othersource()
' ....
'Use the XML from the retrieve to build the input data for the
delete.
If Not oCall.bInitializeMerge(sXML, "phone", "saveCollection",
"", "", sMethodStatus) Then
'Error out
End If
If Not oCall.bGetRowset("phones", oRst, sMethodStatus) Then
'Error out
End If
While Not oRst.EOF
'Step through the rowset looking for the phone we want to update
```

```
If oRst.Fields("primaryId") = sId Then
'Tell OTM to delete this object.
'You could also remove the object properties not needed for a delete action
' before sending the data back to the server, but this is not necessary.
' The extra properties will just be ignored.
If Not oRst.bSetRowProps("delete", "partial", sMethodStatus) Then
'Error out
End If
oRst.MoveNext
Else
'To save OTM some work, strip out all the previously retrieved phones
'from the collection, except for the one to be deleted.
'This does not delete them from the database, only from the working rowset object.
If Not oRst.bRemoveRow(sMethodStatus) Then
'Error out
End If
End If
Wend
If Not oCall.bExecuteLocal(sStatusType, sMethodStatus) Then
'Error out
End If
End Sub
```

Combined Collection Operations

You can combine inserts/updates/deletes in a single saveCollection call. The following code fragment shows how to update one object in a collection item while deleting another.

```
' sDeleteId is the primaryId of a phone to delete
' sUpdateId is the primaryId of a phone to update
If Not oCon.bCreateLBOCall(oCall, sMethodStatus) Then
```

```
' Error out
End If
' Use the XML from a retrieve to build the save.
If Not oCall.bInitializeMerge(sXML, "phone", "saveCollection",
"", "", sMethodStatus) Then
' Error out
End If
If Not oCall.bGetRowset("phones", oRst, sMethodStatus) Then
' Error out
End If
While Not oRst.EOF
' Step through the rowSet looking for the phones identified
above.
Select Case oRst.Fields("primaryId")
Case sDeleteID
' Found the one to delete. Update the action attribute.
If Not oRst.bSetRowProps("delete", "keysOnly", sMethodStatus)
Then
' Error out
End If
oRst.MoveNext
Case sUpdateId
' Found the one to update. Add the new value(s).
oRst.Fields("phoneNumber") = "4255551212"
' Update the action attribute..
If Not oRst.bSetRowProps("update", "all", sMethodStatus) Then
' Error out
End If
oRst.MoveNext
Case Else
' To save OTM some work, strip out all the previously retrieved
phones
' from the collection, except for the ones to be
updated/deleted.
```

```
' This does not delete them from the database, only from the
working rowset object.
If Not oRst.bRemoveRow(sMethodStatus) Then
' Error out
End If
End Select
Wend
If Not oCall.bExecuteLocal(sStatusType, sMethodStatus) Then
' Error out
End If
```

Customization Guidelines Overview

By following these guidelines, consultants, partners, and customers can:

- Rapidly develop new features and modifications
- Easily identify existing customizations
- Avoid upgrade hassles by following standard methods of addressing customizations

The OEP product CD provide all OEP user interface code, and you can modify the OEP code as needed. Custom code will need to be re-implemented when you upgrade to the next version of OEP, so be sure to place your customizations into a custom directory to facilitate the upgrade process.

Warning: Although it is possible to customize all OEP areas, do not modify UCW-enabled areas without using UCW. UCW configurations are applied automatically when you upgrade to new versions of OEP. Non-UCW customizations are not applied automatically when you upgrade and can cause unpredictable results.

Onyx Code Libraries

The OEP folder tree contains a number of files that you can leverage for use in customizations. The more commonly used objects and functions are documented in the Programmer's Reference. Other useful utility functions are stored in files that can be included in ASP pages when needed.

Programmer's Reference

The objects and functions described in the Programmer's Reference provide reusable and robust code that is used throughout the OEP application. By incorporating these items into your customizations, you can quicken your development rate and assure compatibility with future versions of OEP. The following table lists the features documented in the Programmer's Reference and the uses they provide.

Feature	Use
OTMHelper classes	Abstracts interactions with the Onyx Transaction Manager (OTM) of the OEAS.
Cached data	Maintains session and application state for both ASP and the OEAS.
Security	Abstracts interactions with the security features of the OEAS.

Common code

Many other common functions that are shared among the OEP feature set are available for use within your customizations. These functions are stored in a number of source files within the website folder tree. The functions can help with many of the more mundane tasks of generating HTML and configuring user actions within browser clients.

HTTP headers

Two files located in YourOEPwebsite\common\include folder contains code that sets the HTTP response headers to configure client and proxy caching.

File	Usage
httpheaders.asp	Sets response headers to prevent the ASP-generated data from being stored in the cache of the client browser or a proxy. This prevents a proxy from storing the page and forces the browser to request the data from the server every time the page is called.
httpheaderscache.asp	Sets response headers to limit a proxy to cache the page for a single client. Also instructs the browser to cache the page for a maximum of 12 hours.

Other functions

The following tables list some of the other useful functions and summarize their features. All folder paths are listed relative to the root path of the OEP virtual application. You can view the functions by parent file or by category. Review the code of each function to learn more about how they can be added to your custom code.

You don't have the right version of the MSXML DOM here.

Implementation Tips

This topic provides several suggestions for implementing customizations on an OEP system. By following a few simple steps to maintain consistency across customizations, you can simplify your development and maintenance efforts to accelerate the pace at which you add new features to OEP.

Don't customize UCW-enabled areas

Use UCW to configure UCW-enabled areas in OEP. Don't write custom code directly to these areas of OEP because the custom code is not automatically applied when you upgrade from this version of OEP to the next. Also, customizing UCW-enabled areas directly likely causes unpredictable results.

Use UCW tools wherever possible

UCW enables you to configure aspects of UI appearance and behavior that previously could be accomplished only by writing custom code. Furthermore, UI configurations that you implement with UCW can be upgraded from one version of OEP to the next without requiring you to re-implement them.

Use custom folders

Create a new folder for custom HTML and ASP pages at the root of the YourOEPwebsite directory. By doing this, customizations are as isolated as possible from the standard Onyx pages. Onyx recommends that if you need to modify standard OEP files, copy them to a separate directory and modify them there, leaving the original files in the YourOEPwebsite directory. If you do this, you must also modify other OEP files to point to the custom version in the separate directory.

> **Important:** Any custom file that is used by both UCW-enabled areas and non-UCW-enabled areas should be placed in a YourOEPwebsite\ucf\data\custom\{your company name} folder.

Customizations that exist in their own folder structures will not be affected by setup programs that update the system when OEP is upgraded. By reserving your own space for code files, you can migrate your customizations to the newer version with a minimum of hassle.

Use custom identifiers

Tag your customizations with CS, or any kind of indicator that flags them as a custom addition. For example, if you create a custom function for enforcing mandatory fields, call it vbCSEnforceMandatoryFields rather than vbEnforceMandatoryFields. Use the standard prefix conventions for functions, variables, and files, and add the CS after it.

Comment all customizations

Add a single-line comment to the beginning of each OEP function that you change. Then add a set of comments to the start and end of the changes you make. Use the keywords CSBeginMod and CSEndMod to denote your additional customizations. This way, you can search the entire directory by keyword to find changes. The following example shows this commenting standard.

```
...
<%end if%>
<% ' -- CSBeginMod TF - New Toolbar button -- %>
<% PrintIconEvents "iconCompanyLink" %>
<a href="ASP/WebsiteTB.asp" target="_blank">
<img alt="<%=Request.QueryString("CustomerType")%> Web Site"
border="0" height="20"
src="images/icons/iconCompanyLink0.gif" width="20"
id="iconCompanyLink">
</a>
```

```
</span>
<% ' -- CSEndMod TF - New Toolbar button -- %>
...
```

Comments in all client-side files are downloaded to the client machine and parsed by Internet Explorer. Therefore, you must balance the need for understanding against the need for client browser efficiency.

Use include files

Divide custom code into task-related include files. You can then include your code anywhere in the application as needed. ASP uses the following syntax for including the contents of one file in another:

```
<!-- #include file="../Custom/surveys-details_ext.asp"-->
```

Use resource files

Use resource files to maintain your string data. If you need additional string data in the course of customizing OEP, add them to the resource file associated with the UCW-enabled area you are customizing.

Use relative paths

Always use relative paths when referring to files in your custom directory. This way you can easily move customizations between development and production Web sites without modifications.

Test small changes

By changing small pieces of code and then testing, you decrease the amount of time and complexity of implementing your customizations.

Use the Data Area frame or a separate window

Completely custom interfaces can appear in the OEP Data Area frame or in a separate browser window. Consideration must be given to whether or not the feature's target link is capable of being hosted in the Data Area frame. In the case where a link URL target is not frame-compatible, set the URL target parameter to "_blank". This can be accomplished in a number of different ways depending on how the link is implemented. Two possible methods appear below.

- If the link is implemented in the MetaTreeData table, set the vchTarget value to "_blank".
- If the link is implemented as a toolbar button, set the anchor target parameter to "_blank" as shown in the following example.

<a href="ASP/YahooTB.asp" target="_blank">

Be aware of OEP conventions

When customizing OEP, be aware of the following coding conventions.

Naming conventions for shared functions

Because multiple scripting languages are in use throughout OEP, the Onyx naming convention includes a prefix for function names that identifies the source language. For example, tag VBScript

subroutine and function names with a vb prefix. JavaScript function names are not prefixed but legacy functions were named with the js prefix.

Files cached in client browsers

To maximize performance over low-bandwidth connections, OEP employs a caching strategy designed to reduce the transfer of static data files as much as possible. All files not generated from ASP scripts (for example, those with the following file extensions: .htm, .js, .vb, .gif, .jpg, and .css) are delivered to the browser with a content-expiration header that instructs the browser to cache the file for two days. The majority of dynamically generated content, however, uses content-expiration headers that instruct the browser to not cache the resulting data at all (there are a few ASP scripts use 12-hour headers for content that changes infrequently).

If you make changes to any of the static files on the OEP server (for example, you change button graphics or modify style sheets), these changes may not propagate to all client users for as long as two days. To expedite the distribution of updates, instruct your users to clear their browser caches.

You can also expedite the distribution of changes by updating the content expiration settings on each OEP virtual directory through the Internet Services Manager. This may affect performance on systems with large numbers of users; however, as each client systems will download static files with greater frequency than normal.

Multiple Time Zone Support

To maintain consistent behavior in an environment that may contain client applications from different time zones, OEAS processes all date and time information using Coordinated Universal Time (UTC) and the universal date format (YYYY-MM-DD HH:MM:SS). It is the responsibility of OEP and other client applications to implement a scheme to convert this data on input and output to suit the needs of end-users. UTC, like Greenwich Mean Time, is set at zero degrees longitude on the prime meridian and does not adjust for daylight savings time.

To meet the requirements set by OEAS, OEP contains a set of simple functions that convert the time from the server to be useable in the client, and back again when data is saved. These functions are known as the Multiple Time Zone (MTZ) functions.

Use these functions to do the following:

- Convert time data between UTC and the client's local time zone
- Format date and time values in a manner that the end user expects (based on their locale settings)
- Validate the date and time values typed by users

Important: The OEP server does not implement any functionality to generate date and time information that is compatible with the needs of OEAS or the client browser. You should not generate dates from within ASP scripts running in IIS unless you are intimately familiar with the requirements and consequences.

File setup

The functions are distributed in the following files:

- common\include\DateServerFunctions.asp
- common\javascript\datetime.js

DateClientFunctions.asp includes a reference to DateServerFunctions.asp, so it is not necessary to include both if you need the client side functions.

Since many of the functions provided in these files depend on the context cache for the proper display formats for the client browser, the following files must be included in the page beforehand:

- common\javascript\cache.js
- common\javascript\cached_data.js
- common\javascript\common.js
- common\javascript\alert.js
- res\javascript\alertres.js.
- res\datetime_res.js

Failing to include these files may cause unexpected results.

Function descriptions

Each of the files, along with some of their important functions appears in the following tables. Review the code in each file to learn more about the functions and their behavior.

DateServerFunctions.asp

common\include\DateServerFunctions.asp contains several functions that convert dates from OEAS business object XML to a client-specific format. These are intended for use on the server side.

Function	Description
vbConvertUniversalToLocal	Converts a universal date/time string into a date string that matches the client's locale settings

DateTime.js

common\javascript\DateTime.js contains several client-side functions that convert date and time values between formats and time zones. Most of these functions also handle formatting and will respect the client browser's locale settings.

Function	Description
jsConvertGMTUniversalToLocal	Converts a UTC date in universal format to local date format and local time zone
jsConvertJavascriptDateToOnyxFormat	Converts a JavaScript datetime object to local date format
jsConvertLocalToGMT	Converts a date from local to universal format. Has a boolean argument that indicates whether the value should be converted to UTC.
jsConvertUniversalFormatToLocalFormat	Converts a date in universal format to local date format but does not convert time zones

Function	Description
jsConvertUniversalTimeToLocal	Converts a time from universal format (HH:MM:SS) to local time format and local time zone
jsCurrentClientDate	Creates a string that contains the current date in the client's locale format and time zone
jsCurrentClientDateTimeAdd	Creates a string that contains the current date plus a supplied offset in the client's locale and time zone
jsCurrentClientDateTime	Creates a string that contains the current date and time in the client's locale format and time zone
jsFormatGMTUniversalDate	Accepts the individual numbers of a local date and creates one in universal format and converted to UTC
jsFormatUniversalDate	Accepts the individual numbers of a local date and creates one in universal format, but does not convert to UTC
jsValidateInputDate	Validates the contents of a text box to see if it matches the locale's formatting requirements for a date
jsValidateInputDateTime	Validates the contents of a text box to see if it matches the locale's formatting requirements for a date and time
jsValidateInputTime	Validates the contents of a text box to see if it matches the locale's formatting requirements for a time
jsValidateTime	Validates the contents of a supplied string to see if it matches the locale's formatting requirements for a time
jsXMLDateNodeFormatMTZ	Given an XML DOMDocument and a node name, converts all universal date strings to local format and local time zone
jsXMLDateNodeFormatLocal2MTZ	Given an XML DOMDocument and a node name, converts all local date strings to universal format and UTC
jsXMLUniversalDate2LocalDate	Given an XML DOMDocument and a node name, converts all universal date strings to local format, but does not convert to local time zone
vbLocaleDisplayDateFormat	Generates a string that shows the expected date format for the language locale (for example, DD-MM-YYYY HH:MM)
vbLocaleDisplayTimeFormat	Generates a string that shows the expected time format language locale (for example, HH:MM)

Using Diagnostics

OEP includes a set of diagnostic ASP files that output the contents of the application's caches for review. For security reasons these files are restricted to access only by the Administrator and OEP.Administrator roles.

Note: Onyx Product Support currently does not support the use of diagnostics.

Loading the diagnostic pages

To load the initial diagnostics page, click the Diagnostics link on the Navigation Bar (the default settings on installation provide this link to all users who have the proper permissions), or open a new browser instance and connect to http://YourOEPwebsite/Diagnostics. Connecting to the diagnostics pages by typing the URL in an active OEP session will cause the current session to expire and force the user to login again. Once you are connected, the starting diagnostics page displays general server information. Below the server data are links to different sections of important OEP application data. Information about the links appears in the list below.

Link	Description
Application cache	Lists the contents of the Web server's application cache
Client side diagnostics	Contains links that display different types of data cached on the client
Context cache	Lists the contents of the context cache for the active user ID
Server variables	Lists the complete collection of server variables (from the ASP Request.ServerVariables collection)
Version checks	Lists version information for the OEP server and client software. The table that appears shows the minimum required version and the version currently in use.
Cache flushing	Loads tools for clearing and reloading portions of the system caches
User list	Lists the user IDs that have logged onto the OEP system since the application was last started.
Attachment diagnostics	Indicates whether the basic requirements for the attachment server have been met.

Configuring security

The default security settings limit access to the files by members of the OEP Administrators role. The files are restricted through security resources that are installed during OEAS setup.

The resources that control access to the diagnostics are:

- UI:OEP.diagnostics.frontPage
- UI:OEP.diagnostics.frontPage.pathDetails
- UI:OEP.diagnostics.appCache
- UI:OEP.diagnostics.contextCache
- UI:OEP.diagnostics.serverVariables

- UI:OEP.diagnostics.cacheFlush
- UI:OEP.diagnostics.userList

All of the resources must exist for the default diagnostic page to load without error. You can assign individual user accounts to the resources to grant access to the different information areas.

There are no resource requirements for the version checking tools. This page (versionchecks.asp) can be opened by anyone needing information regarding the versions of server and client software in use.

Flushing the application caches

One of the diagnostic tools enables you to clear the caches on the server as needed. By using the diagnostic tools, you can update the server without having to restart IIS each time you make minor changes to OEP system settings.

Cache flush page

The cache flush page is divided into four sections.

Section one: The first section, in the upper-left corner, contains the URLs of the servers that will have their caches cleared. To add a server, type the server URL and the root directory for OEP. Click  to add the server to the list. Continue adding servers as needed (if you are running in a multiple-server environment). To remove a server from the list, select it and click  .

Section two: The second section, in the upper right, is for user log on. The only username and password that is accepted here is the one that OEP uses to connect to OEAS. This username is specified during setup and stored in otm_security.asp. Type the username and password and click  . A message box that contains result XML appears and explains the success or failure of each cache clearing operation.

Section three: The third section, in the lower right, contains the caches that can be cleared and reloaded. Check the box for each cache to reload.

Section four: The final section, in the lower left, contains the caches that can be flushed. These caches are not automatically reloaded, but are left clear until some portion of the application requests the data. On a busy system this will only be a few minutes at most.

Important: These operations should only take place when the system is not experiencing heavy load, especially in a multiple-server environment. Users should not experience any issues with data or session loss, but for best results restrict use of this tool to times when the number of concurrent users is relatively low.

OTM Logging

OEP contains a built-in tool for logging all business object method calls made through the OTM. Information about each call is written to a pair of text files on the OEP server. Developers can then review these files to monitor data moving in and out of OEAS, and to obtain more information on the number of calls made through the OTM during system operation.

The logging feature creates only a minor impact on OEP performance, but the log files themselves can grow quite large. For example, when OEP completes initialization the size of the details file is over three megabytes.

Setup

As the OTM can be accessed from both the client-side and server-side OTM Helper objects, there are separate files that must be edited for each side to be activated.

The server-side path is YourOEPwebsite/common/include/otm_helper.asp.

The client-side path is YourOEPwebsite/common/include/otm_helper_end_point.asp.

The lines of code to edit appear near the top of each file. They are:

```
class="Code">Const OTM_HELPER_XMLLOG = true

Const OTM_HELPER_XMLLOG_PATH = "c:\otmlogfolder\"
```

Important: The folder information stored in OTM_HELPER_XMLLOG_PATH must identify an absolute path to an existing folder. If either of these conditions is not met, OEP will report an error during system initialization.

By naming different folder paths, or by leaving one side configured not to log, you can separate or limit the information about the calls by side.

Any edits to the otm_helper.asp file will cause the OEP server application to restart. The file should only be edited when the delay of reinitialization won't adversely affect end users.

The files

Logging information is saved according to OEAS session ID. For each logged session there are two files: a calls file and a details file. Filenames are prefixed by session ID. For example, the two files created for the session ID 660019F4-0841-4A11-B265-5617CCE81D7F are:

- 660019F4-0841-4A11-B265-5617CCE81D7F-calls.log
- 660019F4-0841-4A11-B265-5617CCE81D7F-detail.log

The calls file contains summary information for each method call. The file includes the time the call was made, the business object and method executed, and the OEP ASP file that made the call.

```
23-Jul-02 11:31:18  customer         retrieveSummaryInfo
/YourOEPwebsite/incident/Incident_GetContactInfo_xmlhttp.asp

23-Jul-02 11:31:19  externalContact  retrieveListByOwner
/YourOEPwebsite/common/include/otm_helper_end_point.asp

23-Jul-02 11:34:38  notification     getNotificationCounts
/YourOEPwebsite/getnotificationcounts_xmlhttp.asp
```

The details file contains the specifics of each call. This includes the input and output XML and any error information.

```
Timestamp: 23-Jul-02 11:26:26
OEAS Call: systemParameter.retrieve
COM+ call returned: no error
VBScript error: no error
In  XML:
<parameters>...</parameters>
Out XML:
<returnXml>...</returnXml>
Custom data:
<customData></customData>
```

You can delete the files at anytime while the system is active. OEP will recreate them if they are missing.

Customizing the OEP Client

This book suggests several ways in which you can customize OEP by writing custom code.

Tip: Remember to use UCW to modify UCW-enabled OEP areas. Do not write custom code directly to these areas, or unpredictable results may occur.

Heartbeat Customizations

OEP implements a session preservation system that optionally keeps client connections active within the OEAS server. Sessions do not timeout because heartbeat calls that check for new Messenger messages and for reminders at regular intervals. When the OEAS system parameter OEPEnableSessionTimeout is set to N (disable), the heartbeat calls maintains the server session because OEAS considers these calls to be no different from user-initiated calls. This setting allows a client user the freedom to leave OEP open on their desktop for as long as they like without experiencing a session timeout.

When OEPEnableSessionTimeout is set to Y, OEAS differentiates between heartbeat calls and user-initiated calls. When an OEP user has not interacted with OEP within a defined time limit and session timeout is enabled, the OEP session is ended. The GlobalSessionTimeOut system parameter defines the time limit, which has a 20% grace period. For example, if GlobalSessionTimeOut is 600 seconds, the session ends after 720 seconds of inactivity.

By making updates to the JavaScript file *YourOEPwebsite*/common/javascript/notification.js, you can change the interval length and add custom actions to the heartbeat's function.

Interval adjustment

The default heartbeat interval is five minutes, which is half of the length of the default lifespan of an OEAS session. To change the heartbeat interval, update the following line within the notification.js

file:

```
var mlHeartbeatIntervalNormal = 300000;
```

The length is measured in milliseconds, so the above value is equal to five minutes.

To prevent sessions from timing out, the heartbeat interval must be less than the lifespan of an unused OEAS session. The OEAS session size is specified by the system parameter GlobalSessionTimeOut. The default OEAS session timeout is 600 seconds (10 minutes). For timing reasons, it is not wise to change the client heartbeat interval to a value greater than the system timeout.

Adding other checks

When the heartbeat interval expires, the function **jsGetNotificationCounts** is called and checks the server for new Messenger messages, active reminders, and new keywords. The function takes a single Boolean argument, which indicates whether or not a dialog box should appear if new messages are present.

You can update this function and the contents of the server-side script file *YourOEPwebsite*\getnotificationcounts_xmlhttp.asp to provide any functionality that must happen in the background and on a regular basis.

Adding the heartbeat to modal windows

The heartbeat code is initialized when the main OEP window loads. This code remains active during regular operations and maintains the OEAS session automatically. The only time this code does not execute is when an OEP feature opens a modal window or dialog box. Thus, a modal window opened from within OEP must reinitialize the heartbeat code within its own scope. Once the modal dialog closes, the heartbeat within the main frame resumes automatically.

All modal windows that exist as part of the standard OEP code base contain JavaScript code to maintain the heartbeat while they are open. If you add a modal window or dialog box to a feature, you must do the following to maintain the heartbeat:

- Include references to the JavaScript files *YourOEPwebsite*/common/javascript/notification.js and *YourOEPwebsite*/res/common/javascript/notificationres.js.
- Call the function jsInitHeartbeat with the argument 'false' when the modal window loads (passing false prevents Messenger notification from firing while the modal dialog is open).

Note: A modal window is any window that is opened using the showModalDialog method of the DHTML window object. All other windows are modeless and do not require any code to maintain the heartbeat.

Using the UCW ProfileId ID

To retrieve the profile ID of the current user in conjunction with a customization on a non-UCW page, obtain a reference to the Application page via the invokeAction function. The following snippet retrieves the profile ID:

```
profileId = invokeAction("direct", "ucf.global",
"getProfileId");
```

The page that uses invokeAction must include YourOEPwebsite\ucf\infrastructure\dynamic_forms\df_common.js.

Using the Spelling Checker

OEP integrates with Microsoft IE to provide users with a spelling checker for work notes in incident and work ticket edit windows. This functionality is supplied as part of IE

Customizing Page Navigation

OEP's header bar and the navigation bar enable users to navigate to various OEP features. Use the UCW Navigation Designer to configure OEP page navigation.

Customizing Resource Files

OEP uses resource files to store text strings that appear throughout the OEP user interface. These strings include labels for user interface elements, text for mouse-over tooltips, and dialog box contents. These files simplify the task of configuring and customizing OEP.

Tip: To edit text strings for UCW-enabled areas, use the UI Text Editor. Edit resource files only for areas of OEP that are not enabled by UCW.

Each server-side executable script file (.asp, .vb, .vbi, .js) that can display text strings to the end user has a corresponding resource file. With the exception of database-generated error messages, which are stored in the National Language table of OEAS, all OEP client text is stored and editable through resource files (to configure database error messages, use the National Language tool in OES Table Administration).

Resource files for new features are stored in an XML format. All new end-user OEP features use these files and the XResourceManager JavaScript class. All resource strings for these features are handled in the client browser.

Contents of the Resource Files

All strings that appear in the user interface are stored in resource files. The strings are used for the following types of interface elements:

- Window titles
- Descriptive captions
- Image ToolTips
- Messages (primarily errors that occur outside the database)
- Column headings for lists

Resource files from previous versions

The strings are declared as constants in the resource files and applied within the script code. All constants follow a loose naming convention where the prefix “res_” is added to a descriptive name of the item that uses the data. For a column or text box label for example, the text that follows the prefix “res_” is the name of the label itself. If the label is “Name”, then the constant is “res_Name”. For messages, the text that follows the prefix “res_” is a brief description of the message. For example, for the message “Your session has timed out. Please log in again.” the constant is named “res_SessionTimeout”.

Some sample names for resource file constants appear in the following table.

String type	Description of constant name	Example
Label	The name of the label with the prefix "res_"	res_Name
ToolTip	The contents of the ToolTip with the prefix "res_"	res_AddCompany
Message	A brief message description with the prefix "res_"	res_SessionTimeout

XML resource files

Resource strings for newer features, pages that use the result list control, and pages that implement UCW are stored in XML resource files. Strings are manipulated using the XResourceManager object. Strings can also be created dynamically at run time. See the *addString* and *setString* methods to learn how to create resource strings in the client browser.

Tip: The resource strings on UCW-enabled areas should be configured using the UI Text Editor and not with calls to XResourceManager.

The following XML document sample shows the element syntax for the resource strings.

```
<?xml version="1.0" ?>
<strings>
 <string name="field1" id="" text="Contact type" />
 <string name="field2" id="" text="Contact name" />
 <string name="field3" id="" text="Comment" />
</strings>
```

Each string element has three attributes. These are described in the table below.

Attribute	Description
name	A unique identifier for the resource string.
id	Reserved for future use.
text	The text of the resource string. This string can contain tokens that can be replaced at run time.

Location of the Resource Files

During setup, resource files for features from previous versions are installed into the YourOEPwebsite\Res directory. The file structure of the Res folder mirrors the structure of the YourOEPwebsite folder, with files grouped by feature.

The names for resource files are identical to the name of the script file to which the strings belong, but with the suffix “res” appended to the file name. For example, the resource file for workticket_core.vbi is named workticket_core_res.vbi. Every script file that generates user interface text has a corresponding resource file.

Resource files for new features and those that use the result list control and other onyxCommon components are installed in a directory in YourOEPwebsite\ that identifies the language in use. The English version stores the files in YourOEPwebsite\eng. The directory structure for these resource files matches that of the application, but the files themselves are named to reflect their purpose. This was done as the strings may be used by several different script files simultaneously rather than tied to a single script file.

ASP scripts that use resource files reference them with the #include directive. The following example shows how the surveydetails.asp script includes surveys-details_res.asp, and how the reference to a caption string is made within the surveydetails.asp file.

The file surveydetails.asp references surveys-details_res.asp:

```
<!-- #include file="res/surveys-details_res.asp"-->
```

Within the ASP code, the text for the caption of the Survey Question column is taken from the value of the variable:

```
<td class="ColumnHeader"><%=res_Survey_Question%></td>
```

The caption is defined as a constant in surveys-details_res.asp:

```
const res_Survey_Question = "Survey Question"
```

Some client-side files have resource strings stored in JavaScript and VBScript files. The naming conventions are the same as those used for ASP files.

XResourceManager Class

An instance of the XResourceManager class stores the user interface strings for some OEP features. Resource strings can be loaded from files on the server or added to the class dynamically at run time.

To learn more about the properties and methods of the XResourceManager class, see XResourceManager object in the Programmer's Reference.

Customizing Result Lists

OEP uses a result list control to display lists of data retrieved from OEAS. The classes that comprise the control accept XML output directly from OEAS, and then process the XML to display it in user-friendly lists.

By writing a few configuration files and providing a few data formatting functions, you can add the result list control to custom features as needed. The result list control automatically handles almost all user interactions, but it also raises events that you can capture for additional functionality.

The result list control also has features that enable users to configure the lists to meet their needs, such as configuring the order of the list's columns. These changes can be saved to the database and persisted for the user's next session.

Features that use the result list control

The following OEP features use the result list control, which enhances user experience and eases the processing of XML output by OEAS:

- Alternate address list
- Audit log
- Comments
- Email history
- Email thread
- Email links
- History
- Incident lists on the PowerPage
- Scripts list
- Search results
- Subscription review list
- Task lists on the PowerPage and on the incident information window
- Task Manager results

Sample Result List

This example list was created using data taken from a call to the customer.retrieveList business object method.

Some of the items in the data set have been hidden and some appear in the detail area. Click the icon in the second column to reveal the detail area.

You can drag and drop column headings to change their order. You can even move columns between the header row and detail areas.

Click a column heading to sort the list.

List event:

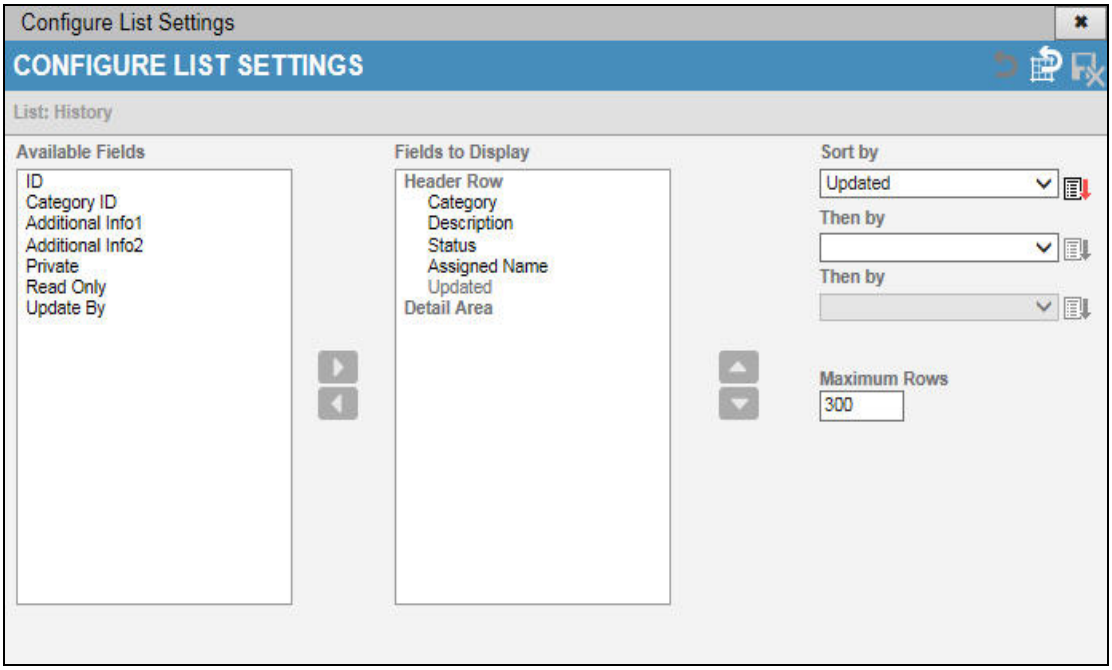
```
Event XML

<eventXML>

 <stuffXML>lalala</stuffXML>

</eventXML>
```

If there is a column where the data is truncated, place the mouse pointer over the data cell and the entire string appears in a tooltip.

Select some of the check boxes to see the selected rows counter (in the bottom center of the list) update accordingly.

Click the page icon in the lower left corner of the list to change the paging mode to single and back again.

On many lists there is a grid icon in the lower right corner that opens the List Configuration Dialog. A screen shot of this dialog appears below. Users can change the placement and visibility of columns, the sorting order, and the maximum row counts. These changes are saved to the database by list name and user name. The changes are automatically reinstated when the user opens the list again in subsequent OEP sessions.



Workflow Overview

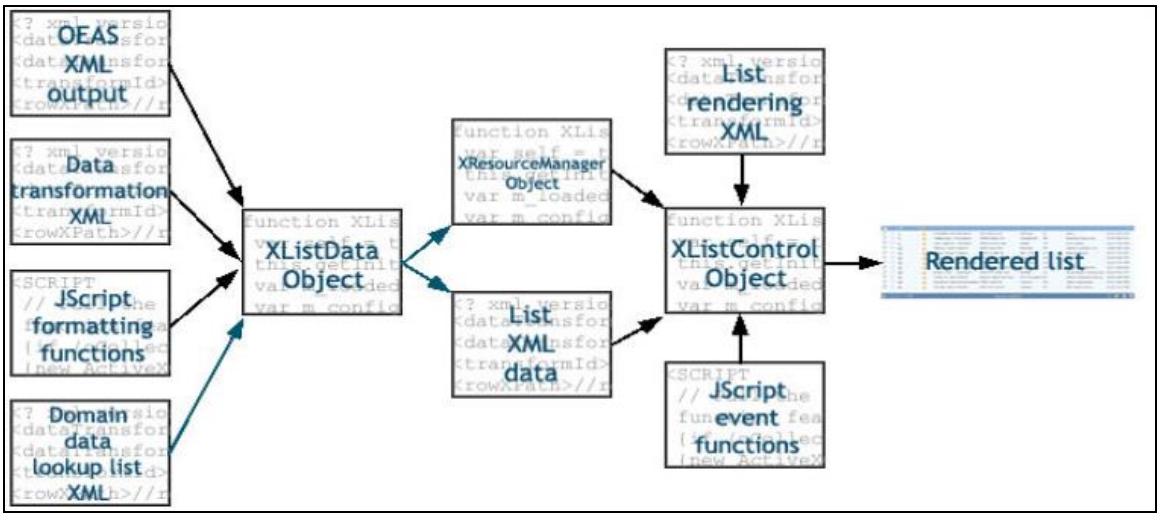
Before data from OEAS can be displayed in an OEP list, configuration data and script code must be prepared for use with the classes that process the data and render the lists. The topics in this section provide an introduction to the overall process and identify the locations on the OEP server where the

list class files are stored. Click a link in the following table to learn more about the process for preparing a list in OEP.

Topic	Description
Implementation diagram	Identifies the pieces that must be assembled before a list can be rendered within the user interface
File set	Identifies the locations on the OEP server where the common list files are stored

Implementation Diagram

OEP renders lists using a combination of JScript classes, XML data files, and JScript functions. The following diagram shows the relationships between these items. The process for rendering a list moves from left to right. Black lines represent required relationships and blue lines represent optional relationships. Click an item in the diagram to learn more about its role in rendering lists in OEP.



Source XML data

This is XML data created from an OEAS retrieveList or retrieveCollection method call. The XListData object can process the data directly from the Onyx Transaction Manager.

Transformation configuration

The configuration XML for the transformation can be stored on the server or generated dynamically at the client at run time. Each configuration has a unique identifier. This allows for storing multiple configurations in a single XML document. See XListData transformation configuration for more information about the contents of the transformation XML.

JScript formatting functions

As all XML data is normally treated as text, these functions exist to enforce formatting rules for data that must be displayed to the user as numbers or dates. The functions must be provided before the transformation can take place. See Data formatting functions for more information about writing these functions.

Domain data lookup lists

As part of the data transformation process, the XListData class can replace domain data identifiers with their human-readable counterparts. The replacement values must be loaded into the XListData class before the data is transformed. See Domain data lookups for more information about creating lookup lists.

XListData object

The XListData object converts data returned from retrieveList and retrieveCollection business object method calls into the format acceptable to the XListControl class. The object requires a valid transformation configuration, and it must have access to the JScript formatting functions that process

the numeric and date-time data in the source XML. See XListData class for more information about using XListData in script code.

XResourceManager object

The XResourceManager object stores the column heading strings for OEP lists. Resource strings can be loaded from files on the server or added to the class dynamically at run time. The XListData object can be configured to automatically create resource strings from a retrieveList data set. See XResourceManager class for more information on using XResourceManager in script code.

List XML data

The data that is used to render the list follows a published format. This data is normally created by the XListData object, but it can be created using any process that formats the data as shown here.

XListControl object

The XListControl object renders a list in the user interface. Before the object is ready it must have a valid rendering configuration, a resource manager object with column heading strings, and event handling functions. See XListControl class for more information on using XListControl in script code.

Rendering configuration

The configuration XML for the rendered list can be stored on the server or generated dynamically at the client at run time. Each configuration has a unique identifier. This allows for storing multiple configurations in a single XML document. See the topic XListControl render configuration for more information about the contents of the rendering XML.

JScript event functions

Each time a user interacts with a list, event data is passed to a handler function for processing. There is no default function for this process. Information about working with events is available in Event handling.

Rendered list

Once all the pieces are in place, the *renderList method* of the XListControl object generates the list HTML and the list appears in the user interface. The JScript event functions alert OEP to events occurring within the list.

File Set

The files OEP uses to render lists in the user interface are stored in the directory *YourOEPwebsite*/onyxcommon/. The following files must be added to a Web page before a list can be rendered successfully:

- xListControl/xListControl.js - Contains the implementation code for the following classes: XListData, XListControl, and XListEvent.
- xListControl/xListControl.css - Contains the style information for the rendered lists.
- xListControl/lcd/lcd.js - Contains the implementation code for the List Configuration Dialog (this file is not necessary if you disable the LCD in the list's render configuration through the allowLayoutPersistence element).

- xListControl/lcd/shared.js - Contains code that is used by both the List Configuration Dialog and XListControl.
- xResourceManager/xResourceManager.js - Contains the implementation code for the XResourceManager class.
- *YourOEPwebsite*/common/javascript/common.js - Contains functions that are used by a list after it has been rendered.

In addition to the required files listed above, the file *YourOEPwebsite*/onyxcommon/common/javascript/xmlCommon.js includes a number of functions that simplify the loading of unparsed XML strings and files into DOMDocument objects.

The client-side scripts that create the list classes must also contain functions for event handling, data retrieval, and XListData transformation data formatting.

Event Handling

A list in a Web browser fires a number of events that can be handled by client-side code. This section explains when list events are generated and the data they carry. Click a link in the following table to learn more about working with list events.

Topic	Description
List events	Identifies and describes each of the events a list may fire
Event XML	Describes the XML data that is passed from the list to the event handling function
Row event data	Describes the XML data that describes the state of certain rows in the list when an event fires

List Events

The following table contains a complete list of the events that may be generated by the XListControl during its lifespan. All events are handled by a single function that is stored in the EventHandler property of the XListControl object.

Event	Fires when
cellLink	The user clicks a hyperlink in a list cell
checkboxSelect	The user checks or unchecks a checkbox for a row
checkboxSelectAll	The user clicks the checkbox in the list header that selects or deselects the checkbox of every list row
dragFieldOver	The user drags a column heading over another column heading
listChanged	The user has made changes to the list. If the changes are made through the List Configuration Dialog, the event does not fire until the user closes the dialog.
onKeyUp	The user has pressed a key while the list has focus
renderComplete	The XListControl has finished rendering the table for the user

Event	Fires when
resetLayout	The user clicks the icon to reset the list settings to the default
resize	The parent HTML element of the list changes in size
rowCollapse	The user collapses the detail area of a row
rowExpand	The user expands the detail area of a row
rowSelect	The user selects a row in the list
rowUnSelect	The user deselects a row in the list
saveLayout	The user clicks the icon to save the current list settings
scroll	The user has adjusted their view of the list by moving one of its scroll bars
sortColumn	The user sorts the list by one of its columns
subFieldSort	The user sorts the list by a column located in the detail area

Each time a list generates one of these events, the XListControl object calls a single function that is assigned to it through the EventHandler property. This function takes two string arguments. The first argument contains the event XML data, and the second contains the name of the event (which is also available in the event XML). The XListControl object does not check the return value of the event handling function.

Although the events represent most of the common actions of the XListControl, these events are raised primarily to allow for other components on the page to react if necessary. The event handler function does not have to help the list perform its basic actions. In many cases the function need not do anything. The event handler function is called after the XListControl completes its own processing of the list event. The function cannot prevent the XListControl from processing an event.

The cellLink event fires when a user clicks a hyperlink in a cell in the list. The event handler function must determine what action results from the link. The "link" is created using a style applied to the text. The browser will not interpret the user's click as a click on a hyperlink.

Event XML

When the XListControl object calls the event handler function, one of the arguments it passes contains an XML data string. The XML identifies the event that was fired and contains information about the current state of the list.

The following fragment shows a sample XML data set for an event. This sample shows all possible elements, although there are few events and list configurations that will result in all elements appearing in the XML with values.

Sample event XML

```
<eventData>
 <checkedRows>3,5,7</checkedRows>
<eventCode>resize</eventCode>
```

```
<eventListId>idActiveList</eventListId>
<eventOtherData></eventOtherData>
<eventRowIndex>-1</eventRowIndex>
<eventType>resize</eventType>
<eventRow>
<row>
 <fieldId>dataValue</fieldId>
  ...
 </row>
</eventRow>
<selectedRow>
 <row>
<fieldId>dataValue</fieldId>
  ...
 </row>
</selectedRow>
 <parentRow>
 <row>
 <fieldId>dataValue</fieldId>
   ...
 </row>
</parentRow>
</eventData>
```

Descriptions for each element in the event XML appear in the following table.

Element	Description
checkedRows	A comma-delimited list of numbers for the rows that have their checkboxes selected.
eventCode	The name of the list event that has fired. Click here to see the complete list of events.
eventListId	The identifier of the list that fired the event. This is taken from the listId element in the XML display configuration.
eventOtherData	Additional event-specific information. See below.
eventRow	A collection of event data fields for the row that fired the event.
eventRowIndex	The data index of the row that caused the event to fire. This element contains -1 if a

Element	Description
	row did not fire the event.
eventType	The DHTML event that caused the list event to fire.
selectedRow	A collection of event data items for the row in the list that is currently selected

The eventRow and selectedRow elements contain the data fields that have been named event parameters in the display configuration. See Row event data for more information about how the data in these elements is organized.

The eventOtherData element contains extra information for the cellLink and listChanged events. For a cellLink event, the eventOtherData element identifies the column (by dataFieldId name) where the user clicked the link. For a listChanged event, the eventOtherData element contains an XML structure that describes the new sorting order for the list.

The XListEvent class provides an interface to parse and extract the event XML data quickly and easily. See the topic XListEvent class for more information on using XListEvent to process event data.

The XListEvent class does not have a mechanism for retrieving the contents of the eventOtherData element. You must use the source event XML to obtain this information.

Row Event Data

Within the event XML for the event handler function, the eventRow and selectedRow elements contain the data fields that have been named event parameters in the display configuration.

eventRow element

This element contains information about the row that fired the event. This element only contains child elements if a row is the source of the event.

selectedRow element

This element only appears in the event XML when a row in the list has been selected by a user. When a user clicks a row to select it, the contents of this element are identical to those in the eventRow element.

Data organization

Each item in the display configuration XML that has its eventParameter element set to true is included as a child of the row elements. Data in each of these elements takes the format shown in the XML fragment shown below.

```
<eventRow>
<row>
<dataFieldId>dataValue</dataFieldId>
</row>
</eventRow>
```

For example, a configuration that had the fields primaryId and secondaryId identified as event parameters would generate the following event XML for the event row.

```
<eventRow>
<row>
<primaryId>88d969c5-f192-11d4-a65f-0040963251e5</primaryId>
<secondaryId>456773</secondaryId>
</row>
</eventRow>
```

Use the *getField method* of the XListEvent object to obtain these data items from the event XML. Fields are labeled with the value of their dataFieldId elements from the display configuration.

Class Overview

There are five classes that are used to render lists in OEP. Each object and its description appears in the following table. Click a link to learn more about the object's role within OEP lists.

Class	Description
XListControl	Renders lists within the browser
XListData	Converts data from retrieveList and retrieveCollection data sets into a format usable by XListControl
XListEvent	Parses event XML data generated by XListControl

XListControl Class

An instance of the XListControl class renders a list in the user interface. The format and properties of the list are determined by a combination of class settings and XML configuration data. The XML data is usually loaded from the Web server but can be supplied dynamically at run time.

The XML configuration settings contain information about the general format and actions of the list and about the display settings for the individual data columns. Some of the settings that the configuration determines include:

- Whether users can select and deselect individual rows in the list
- Which data columns appear in the main list, are available in detail areas, or are hidden completely from view
- Which data columns to include as function parameters for event handlers
- Which data columns can be used to sort the list (and how their data is interpreted as criteria for the sort)
- How data is formatted within columns (justification, text wrapping, column widths, etc)

After the configuration is loaded, the *renderList method* generates the list HTML and places it within a DIV element on the user interface. Users can then interact with the list as needed. After the list is

rendered, it fires events as the user clicks elements within the list. To learn more about writing code to handle XListControl events, see the topic Event handling.

Information about the configuration implementation for the XListControl display XML is available in the topic XListControl render configuration.

The constructor for this class has four arguments, only the first of which is required. The first argument is the JavaScript object variable name (passed as a string). This string is then used within the HTML that is created when the list is rendered. Failure to initialize the object properly causes an error message to appear.

The second argument is reserved for future use and should always be passed as false.

The third argument identifies the language in use for the strings that appear in the list. The default value for this is "eng" for English.

The final argument identifies the path to the internal resource files for the list control. The default value for this is one level below the relative path of the feature using the control.

A sample use of the constructor appears in the following code fragment. The last three arguments are optional. This sample shows their default values.

```
var oListControl = new XListControl("oListControl", false,
"eng", "../");
```

To learn more about the properties and methods of the XListControl class, see XListControl object.

Input Data Specification

While the XListData class can be used to automatically convert a retrieveList or retrieveCollection data set into format for the XListControl class, it is possible to move custom data directly into an XListControl object. To do so, the data need only follow the format shown below:

```
<listData>
 <rows>
 <row>
  <rowIndex>1</rowIndex>
   <c0>Column data</c0>
   <c1>Column data</c1>
   ...
  <cN>Column data</cN>
 </row>
 </rows>
</listData>
```

Because the order of the items in the list may change after initial rendering, some operations require the use of the <rowIndex> element to identify rows from the table. The *getRowData* and *selectRow* methods accept parameters that select rows based on the value of their rowIndex elements.

The first row element has a rowIndex value of one. The column elements begin at zero.

Using Icons in List Columns

You can use icons in list columns to represent data to users graphically. Result lists support two types of icon columns, one for two possible states (Boolean values) and the other for an unlimited number of states.

Boolean values are represented by a check icon. A true value is represented by ✓. A false value is represented by an empty column cell.

Note: For a Boolean column in the XListControl, a value of 1 is true and any other value is considered false.

You can also create your own icons to work with domain data values. The XListControl class uses a convention where the icon file name is a concatenation of a relevant prefix (stored in the configuration file) and the domain data value. When the table is rendered, the filenames are generated by combining the prefix with the data value.

For example, the priority icons used in the email history lists are named priority_0.gif, priority_1.gif, and so on. The prefix stored in the configuration file is priority_. The possible domain values are 0, 1, and 2.

Headings for icon columns

Icon and boolean columns are restricted to a narrow width and cannot be resized by the user. Because of their size limitations, these columns cannot have standard text headings. They can, however, have icon headings with tooltips. The headerImageSource element of the XListControl render configuration contains the info for the graphic that appears in an icon column heading. The tooltip text (and the text that appears in the List Configuration Dialog) is obtained from either the captionAuto or captionResId elements.

Important: If no icon information is set for a column header, the user will not be able to see the tooltip or sort the list by the column's contents. You must at least use a transparent graphic to provide the user with this functionality.

Icon requirements

The XListControl sizes icon columns based on a graphic size of 17 pixels square. The files must be in GIF format.

There must be a file on the server for each and every domain value. There is no option to not display an icon for certain values or to display one icon for a specific range of values. If you want the cell in a column to appear empty for a domain data value, create a blank icon with a transparent background.

Configuring a result list to use icon columns

Create icons according to the conventions listed above. Using the list of available domain values, select a prefix and name the files accordingly.

In the render configuration XML file, set the bodyImageSource element identical to the file prefix. To include an icon column heading set a value for the headerImageSource element. When the user places the mouse pointer over the column heading icon, a tooltip appears with the column heading text. Depending on your list configuration, the tooltip text is either auto-generated or copied from the captionResId element. This text is also used for the column in the List Configuration Dialog.

In the JavaScript code that prepares the table, set the *IconFilePath* property to the directory where the graphic files are stored.

When the table is rendered, the control replaces the data values with the appropriate icons.

Handling Page Resize Events

A result list on page does not automatically refresh when its parent DIV element is resized. The DHTML onresize event requires that the list control have its setListHeight method called to guarantee that the contents of the list remain visible. The easiest way to do this is to set the height equivalent to the offetHeight of the parent DIV element. For example, the following code can be used to refresh the list when the page is resized.

```
var iHeight = document.getElementById
("listContainerDiv").offsetHeight;

oXListCtrl.setListHeight(iHeight);
```

Important: This event code should not be used for the XListControl resize event discussed in the topic List events. Doing so will cause an infinite loop (the XListControl fires its own resize event after the setListHeight method is called).

Resource Files

Strings for column headings are manipulated by using the XResourceManager object. Each XListControl instance of a list must have a resource manager object attached to it for the column headings to appear in the list. These strings can be loaded from a file on the server or created dynamically at run time.

Almost all OEP lists that use XListControl have their column headings generated automatically from the data provided by OEAS. See Auto-generating column headings for more information on doing this with custom lists.

Once the list has been rendered, changes to the strings in the resource manager have no effect on the rendered list.

XListData Class

An instance of the XListData class converts data returned from retrieveList and retrieveCollection business object method calls into the format acceptable to the XListControl class.

How a data set is converted is determined by XML configuration settings. These settings can be stored in files on the Web server or generated on the client at run time. The configuration determines the following:

- Which data columns to include in the list data
- Which data columns require special formatting (for example, numbers with decimal places and dates)
- Which data columns should have numeric identifier data replaced with human-readable data
- Which data columns (for retrieveList data sets) should have their database column information copied to a resource manager object

After the configuration is loaded, the *transformData* method converts the source XML into the XListControl format. The *Output* property contains the finished XML.

Information about the configuration implementation for the XListData conversion XML is available in the topic XListData transformation configuration.

To learn more about the properties and methods of the XListData class, see XListData object.

Domain Data Lookups

As part of the data transformation process, the XListData class can replace domain data identifiers with their human-readable counterparts.

Each instance of an XListData class has an internal object named 'lookup' that stores the replacement values for domain data identifiers. This object accepts XML data in the format shown in the following sample.

```
<listLookup>
 <lookupId>lookup list identifier</lookupId>
<items>
 <item>
 <id>domain-data identifier</id>
<desc>human-readable description</desc>
 </item>
 </items>
</listLookup>
```

The value of the lookupId in the above sample is then referenced by the lookupName element within the data transform configuration. During the data conversion, values in the source data column are replaced with values from the lookup list

For more information about adding lookup lists to an XListData object, see XListData object in the Programmer's Reference.

Data Formatting Functions

As part of the data transformation process, the XListData class can format data for specific regional or presentation requirements. Data columns are assigned a format in the data transformation configuration and JavaScript functions in the client use this information to modify the data for eventual presentation in the rendered list.

Each column in the data transform configuration has a formatType element that determines what sort of processing, if any, the data should be subjected to during the transformation. The supported elements and their descriptions appear in the following table.

Value	Description
boolean	Used for values that are 1 or 0
custom1	One of two available custom formatting options
custom2	One of two available custom formatting options
date	Used for date-only values
dateTime	Used for date and time values
float	Used for floating point numbers
integer	Used for integer values
text	Used for standard text values
time	Used for time-only values

During the data transformation, an XListData object checks the formatType of the column and calls a function associated with the formatting type. There are no default functions for this process. They must be added to the code through a <SCRIPT> element on the page that hosts the list class script code. Failure to do so will result in an error during the data transformation.

At least two functions are required. One is used for date values and the other for numeric values. All columns with a formatType of date, time, or dateTime are routed through one function, and all values of float or integer are routed through a different function. Each custom type can be assigned its own function if necessary. Columns with a formatType of text are copied to the list data output without modification. Columns with a formatType of boolean are cleaned (values of '1' are left intact and any other value is converted to '0') and then copied to the list data output.

There is no special formatting consideration for currency values. If you require such formatting, use one of the custom options and create your own functions.

Writing the formatting functions

The number function that processes float and integer values accepts three arguments. The first argument contains a string of the number data that is to be formatted. The second argument is a Boolean that indicates whether the number is to be treated as an integer. The third argument, which is only included if the second argument is false, indicates the number of decimal places to which the value should be limited. This third argument is taken from the decimalPlaces element of the data transform configuration.

The date-processing function accepts two arguments. The first argument contains a string of the data that is to be formatted. Date information from OEAS will almost always be in universal format (for example, 2001-05-10 10:04:43). The second argument is an integer that indicates the type of data that should be returned. If the argument is 0, the returned value should contain the date and time. If the argument is 1, the returned value should only contain the date. If the value is 2, the returned value should only contain the time.

Functions for processing custom data accept two arguments. The first contains the data to process, the second contains the value of the customParam element from the data transform configuration.

All functions return a string that contains the processed data. This string is inserted into the transformed XML.

Function scope

An XListData object accesses these functions through global variables that are declared in XListControl.js. Custom code that manipulates the lists must assign function references to these variables before attempting to transform list data. The names of the variables appear in the following table.

Variable name	Purpose
mlistDT_ CustomFunction1	Used for custom formatting (formatType of custom1). Does not need to be set if no custom formatting is necessary.
mlistDT_ CustomFunction2	Used for custom formatting (formatType of custom2). Does not need to be set if no custom formatting is necessary.
mlistDT_ DateFunction	Used for date and time formatting
mlistDT_ NumberFunction	Used for integer and float formatting

Auto-generating Column Headings

XML data sets from retrieveList business objects contain elements that describe the column data (data types, string lengths, etc). One of these elements contains a simple description that can be used as a column heading. There are column settings in both the XListData transformation configuration and the XListControl render configuration that cause the objects to copy the retrieveList column information for display in the rendered list. During processing in the client browser, the XListData object copies the column headings to an XResourceManager object. The resource manager is then attached to the XListControl, which extracts the resource strings for use in the user interface.

To configure a list for auto generation of column headings, do the following:

- In the XListData transformation configuration, set the useHeading element to '1' for each column that will have its heading information copied for possible use in the list.
- After the configuration is loaded into an instance of the XListData class, use the *appendResources* method to copy the headings to a resource manager.

- In the XListControl render configuration, set the captionAuto element to '1' for each column that will have its heading information copied from a resource manager and displayed in the list.
- Use the *attachResource* method to add the resource manager with the column heading data to the XListControl instance.

When the list is rendered, the column heading information stored in OEAS appears in the list heading.

Note: Column headings for retrieveCollection methods cannot be auto generated. They must be added to a resource manager and referenced in the XListControl render configuration.

XListEvent Class

An instance of the XListEvent class aids in the processing of events fired by a rendered list on an OEP page. The class is intended for use within the event handling and data retrieval functions that must be attached to the XListControl object before a list can be rendered.

The class parses the event XML and provides methods for simple retrieval of the event data.

The event XML can be used as an argument for the class' constructor. An instance of the class can be reused with different event XML through the setEventData method. Examples of both appear in the following code fragment.

```
var oListEvent = new XListEvent(psEventXML);
...
// Reset the event data
oListEvent.setEventData(psNewEventXML);
...
```

To learn more about the methods of the XListEvent class, see *XListEvent object* in the Programmer's reference.

To learn more about list events in general, see Event handling.

Editing Configuration Files

Data configuration for an instance of the table class takes place in two stages. The first stage selects the data columns that will be used in the table. This stage also converts the data set into a format that the list class can understand.

The second configuration file specifies the initial display format for the data in the list. This stage also determines which data columns are passed to the event handler functions when a user interacts with a list.

Maximum rows settings are stored in a configuration file at the root of the application. Edit this file to limit the number of records that users can return from the database.

Click a link in the following table to learn more about the format of a configuration file.

Topic	Description
XListData transformation configuration	Describes the format for data transformation configurations
XListControl render configuration	Describes the format for list rendering configurations
Maximum rows settings	Describes the file that stores information for default maximum row retrieval settings

XListData Transformation Configuration

The first step in preparing a data set for use with the XListControl object is identifying which portions of the data will be used in the table. Most of the data is selected to be displayed to the user, but other portions may be hidden from view and used for other actions within the user interface (for example, for other database record retrievals).

The configuration information for data selection is stored in an XML file. This information is used by the XListData object when processing data sets. The XML structure of the configuration file identifies the data portions to use in the table and includes information on any possible processing.

Throughout the OEP file set, the data transformation configuration files follow a convention where the feature name is paired with suffix of _data_config.xml. For example, the file for the audit log list is named auditlog_data_config.xml.

The XListData object is configured to look for a specific dataTransform element (identified by child transformId element) within the configuration XML.

Click an element in the sample configuration to learn more about what it does.

```
<dataTransforms>
 <dataTransform>
  <transformId>incidentSales1</transformId>
  <rowXPath>/returnXml/parametersReturn/rowSet/rows/row</rowXPath>

<definitionXPath>/returnXml/parametersReturn/rowSet/columnDefinitions</definitionXPath>
  <columns>
   <column>
    <sourceColumn>c0</sourceColumn>
    <description>Product Number</description>
    <useHeading>1</useHeading>
    <sequenceNumber>0</sequenceNumber>
    <formatType>text</formatType>
```

```
        <customParam />
        <decimalPlaces />
        <lookupName />
        <sourceCustomXSL />
      </column>
    </columns>
  </dataTransform>
</dataTransforms>
```

dataTransform Element

The dataTransform element contains the specification for a data set to be converted to a table class friendly format. The element has four children, each of which are described here.

Element	Description
columns	Contains the data elements that will be used in the rendered list
definitionXPath	An XPath statement that can locate the columnDefinition element of a retrieveList data set
rowXPath	An XPath statement that can locate the individual rows of the data set
transformId	A unique identifier for the transform configuration

definitionXPath Element

The definitionXPath element of the XListData configuration contains an XPath statement that can locate the columnDefinitions element in a retrieveList XML data set.

Legal values

Any valid XPath statement.

Required

No. The element has no default value.

Remarks

The XListData objects uses this information when automatically generating strings for column headings in the rendered list. The *appendResources* method uses this information when creating the strings.

This element has no use with a retrieveCollection data set.

rowXPath Element

The rowXPath element of the XListData configuration contains an XPath statement that can locate the row-level data within the source data set.

Legal values

Any valid XPath statement.

Required

Yes.

Remarks

For a retrieveList data set this element should reference the first <row> element. For a retrieveCollection data set this element should reference the first element within the collection.

transformId Element

The type element of the XListData configuration contains the unique identifier for the transform configuration.

Legal values

The identifier must be limited to alphanumeric characters. No white space or punctuation characters are allowed.

Required

Yes.

columns Element

The columns element contains the specification for the data to be used in the table. Each child column element specifies a data column to copy to the source data XML that will be used in the rendered list. The following table lists the elements that can appear in the child column elements.

Element	Description
customParam	Specifies a custom parameter for a custom formatting function
decimalPlaces	Specifies the number of decimal places to display for floating-point data
description	Contains human-readable information about the field
formatType	Indicates how the source data from the column will be converted during the transform
lookupName	Identifies a lookup table for ID/description replacement
sequenceNumber	Specifies the order in which the data is added to the transformed XML
sourceColumn	Identifies the source element for the row data
sourceCustomXSL	Contains a XSLT statement to extract more complex data from the source
useHeading	Indicates if the transform should use the columnDefinition data for a column heading

customParam Element

The customParam element of the XListData configuration contains data that can be passed to a custom formatting function.

Legal values

Any legal XML text.

Required

No. The element has no default value.

Remarks

How this element can be used with the formatting of list data is explained in the topic Data formatting functions.

decimalPlaces Element

The decimalPlaces element of the XListData configuration specifies the number of decimal places that numeric values should be limited to.

Legal values

An integer.

Required

No. If omitted, the default value of the element is negative one (-1) and decimal values are rendered without truncation.

Remarks

This element is only used for data items of formatType float.

description Element

The description element of the XListData configuration contains human readable text to describe the purpose of the column.

Legal values

Any legal XML text is allowed.

Required

No. The element has no default value.

Remarks

This element is not read by the XListData class. It is provided as an aid to the developer.

formatType Element

The description element of the XListData configuration .

Legal values

'text', 'boolean', 'date', 'dateTime', 'integer', 'float', 'time', 'custom1', and 'custom2'.

Required

Yes.

Remarks

See Data formatting functions for more information on how the content of this element is used.

lookupName Element

The lookupName element of the XListData configuration contains the identifier of a lookup list to replace domain data numeric values with human-readable text.

Legal values

Any valid XML text.

Required

No. The element has no default value.

Remarks

See the topic Domain data lookups for more information on creating lookup lists.

sequenceNumber Element

The sequenceNumber element of the XListData configuration indicates the position of the data field column within the transformed XML.

Legal values

Any integer value greater than or equal to zero (0). Values must be sequential.

Required

Yes.

Remarks

The numbers are paired with the letter 'c' in the data XML used by an XListControl object. More information about the formatting of the output data is available in the topic Input data specification.

sourceColumn Element

The sourceColumn element of the XListData configuration identifies the column from the source data to copy to the list data.

Legal values

Any valid XML text that can identify a column name.

Required

See below.

Remarks

Each column element must contain either this element or the sourceCustomXSL element. If both elements contain data, the sourceCustomXSL element takes precedence.

sourceCustomXSL Element

The sourceCustomXSL element of the XListData configuration contains executable XSL code that can dynamically create row data during the transform.

Legal values

Any valid executable XSL code.

Required

See below.

Remarks

Each column element must contain either this element or the sourceColumn element. If both elements contain data, this element takes precedence.

The XSL code in this element executes within the context of the row element path defined in the rowXPath element.

The XSL code in this element must be escaped before it can be processed successfully by the XListData object.

In the following sample fragment, the sourceCustomXSL element uses the value of the c2 column to determine where the source data is copied from.

This code checks the value of the row's c2 element to see if it is less than ten. If so, the data from column c7 is used as the source. If the value of c2 is greater than or equal to 10, the data from column c8 is used instead.

```
<xsl:choose>
<xsl:when test="c2 &lt; 10">
 <xsl:value-of select="c7" />
 </xsl:when>
<xsl:otherwise>
  <xsl:value-of select="c8" />
</xsl:otherwise>
 </xsl:choose>
```

To prevent the XSL from being parsed as part of the configuration data, it must be preserved by either escaping portions with entity references or by placing the code within a CDATA section. The following fragment shows the above sample escaped using both methods. Note the double escaping used in the xsl:when statement for the less-than test when using entity references.

```
<!-- First with entity references -- >
<sourceCustomXSL>
 &lt;xsl:choose>
&lt;xsl:when test="c2 &amp;lt; 10">
  &lt;xsl:value-of select="c7" />
 &lt;/xsl:when>
 &lt;xsl:otherwise>
&lt;xsl:value-of select="c8" />
 &lt;/xsl:otherwise>
 &lt;/xsl:choose>
</sourceCustomXSL>

<!-- Second, within a CDATA section -->
<sourceCustomXSL>
<![CDATA[
 <xsl:choose>
 <xsl:when test="c2 &lt; 10">
<xsl:value-of select="c7" />
 </xsl:when>
<xsl:otherwise>
<xsl:value-of select="c8" />
</xsl:otherwise>
 </xsl:choose>
]]>
</sourceCustomXSL>
```

useHeading Element

The useHeading element of the XListData configuration indicates whether the object should copy the column heading information to an XResourceManager object automatically.

Legal values

Zero (0) for false and one (1) for true. Any other value is interpreted as false.

Required

No. If omitted, the default value of this element is zero (false).

Remarks

This element is only useful when working with retrieveList data sets.

This element cannot be used with column data generated by the sourceCustomXSL element.

XListControl Render Configuration

The render configuration normally appears in the file listDataConfig.xml. A path to this file can be set through a property of an XListControl object, or the object can be configured using a MSXML DOMDocument object at runtime. Like the XListData transformation XML, multiple configurations can be stored in one file. The objects then locate the appropriate configuration by searching on a unique identifier.

Throughout the OEP file set, the render configuration files follow a convention where the feature name is paired with suffix of _list_def.xml. For example, the file for the audit log list is named auditlog_list_def.xml.

A complete minimum sample configuration appears below. Click an element in the XML to learn more about its role in the configuration.

```
<listConfiguration>
 <listDefs>
  <listDef>
   <listId>incident1</listId>
   <description>The first list for incidents</description>
   <allowDetailArea />
   <allowDetailAreaSort />
   <allowDragDrop />
   <allowLayoutPersistence />
   <allowRefresh />
   <allowSelect />
   <allowUnselect />
   <allowUserColumnSizing />
   <initHeartBeat />
   <maxRowsMax />
   <minMaxRows />
```

```
        <pageView />
        <rowIndicatorState />
        <showRowCount />
        <showSelected />
        <dataFields>
         <dataField>
          <dataFieldId>myfield1</dataFieldId>
          <type>column</type>
          <dataSourceColumn>c0</dataSourceColumn>
          <captionAuto>1</captionAuto>
          <sequenceNumber>1</sequenceNumber>
          <cellType>text</cellType>
          <defaultColumnWidth>170</defaultColumnWidth>
          <eventParameter>1</eventParameter>
          <alternateSortField />
          <bodyImageSource />
          <captionResId />
          <cellWrap />
          <columnStyle />
          <headerImageSource />
          <initialSortDirection />
          <justify />
          <maxColumnWidth />
          <minColumnWidth />
          <sortDataType />
          <sortSequence />
          <userSortable />
         </dataField>
        </dataFields>
       </listDef>
      </listDefs>
     </listConfiguration>
```

listDef Element

The listDef element contains a single specification for a list. The element can have up to 18 children, each of which is described in the following table.

Element	Description
allowDetailArea	Indicates if users can view and move columns to a detail area
allowDetailAreaSort	Indicates if users can sort the list by columns in the detail area
allowDragDrop	Indicates if users can drag and drop columns to change their position in the list
allowLayoutPersistence	Indicates if users can save their changes to a list
allowRefresh	Indicates if users can refresh the contents of a list
allowSelect	Indicates if users can select rows in the list
allowUnSelect	Indicates if users can unselect rows in the list
allowUserColumnSizing	Indicates if users can change the size of list columns
dataFields	Information about the data fields that are used within the list
description	Human-readable text about the use of the configuration
initHeartBeat	Indicates if the list configuration dialog will maintain the system heartbeat while it is open
listId	A unique identifier for the list configuration
maxRowsMax	Contains the highest value a user can specify as a maximum rows limit
minMaxRows	Contains the lowest value a user can specify as a maximum rows limit
pageView	Indicates whether the list should display all rows at one time, or divide the list into user-selectable pages
rowIndicatorState	Identifies the current row highlighting setting
showRowCount	Indicates whether a counter should display how many rows are in the list
showSelected	Indicates whether a counter should display how many of the rows are currently selected

allowDetailArea Element

The allowDetailArea element of the XListControl configuration indicates whether the list will produce a detail area.

Legal values

Zero (0) for false and one (1) for true. Any other value is interpreted as true.

Required

No. If omitted, the default value is one (true).

Remarks

If this element is false, any other settings that indicate a detail area should be present are ignored. Users will not be able to create a detail area using the LCD.

allowDetailAreaSort Element

The allowDetailAreaSort element of the XListControl configuration indicates whether the users can sort the entire list by clicking a heading for a column in the detail area.

Legal values

Zero (0) for false and one (1) for true. Any other value is interpreted as false.

Required

No. If omitted, the default value is zero (false).

allowDragDrop Element

The allowDragDrop element of the XListControl configuration indicates whether users will be allowed to move columns around by dragging them with the mouse.

Legal values

Zero (0) for false and one (1) for true. Any other value is interpreted as true.

Required

No. If omitted, the default value is one (true).

allowLayoutPersistence Element

The allowLayoutPersistence element of the XListControl configuration indicates whether users will be allowed to save changes to the table layout.

Legal values

Zero (0) for false and one (1) for true. Any other value is interpreted as true.

Required

No. If omitted, the default value is one (true).

Remarks

If this element is false users will not be able to view the LCD. Other changes can still be made directly to the list (drag and drop columns, resize columns, and sort actions) but users will not be able to save their changes.

allowRefresh Element

The allowRefresh element of the XListControl configuration indicates whether users will be able to refresh the list contents by clicking an icon on the right side of the bottom list border.

Legal values

Zero (0) for false and one (1) for true. Any other value is interpreted as true.

Required

No. If omitted, the default value is one (true).

Remarks

If this element is false, the icon for refreshing the list contents is hidden.

allowSelect Element

The allowSelect element of the XListControl configuration indicates whether users will be allowed to select individual rows in the list.

Legal values

Zero (0) for false and one (1) for true. Any other value is interpreted as true.

Required

No. If omitted, the default value is one (true).

allowUnSelect Element

The allowUnSelect element of the XListControl configuration indicates whether users will be allowed unselect individual rows in the list.

Legal values

Zero (0) for false and one (1) for true. Any other value is interpreted as true.

Required

No. If omitted, the default value is zero (false).

Remarks

If this value is false, table rows are unselected only when the user clicks a different row in the list.

allowUserColumnSizing Element

The allowUserColumnSizing element of the XListControl configuration indicates whether users will be allowed to change the width of list columns dynamically.

Legal values

Zero (0) for false and one (1) for true. Any other value is interpreted as true.

Required

No. If omitted, the default value is one (true).

dataFieldId Element

The dataFieldId element of the XListControl configuration contains a unique identifier for the field within the list.

Legal values

The identifier must be limited to alphanumeric characters. No white space or punctuation characters are allowed.

Required

Yes.

Remarks

The value of this element must be unique among the columns within a single list configuration.

description Element

The description element of the XListControl configuration contains human-readable text to describe the purpose of the list.

Legal values

Any legal XML text is allowed.

Required

No. The element has no default value.

Remarks

This element is not read by the XListControl class. It is provided as an aid to the developer.

initHeartBeat Element

The initHeartBeat element of the XListControl configuration indicates whether the List Configuration Dialog will maintain the system heartbeat while it is open.

Legal values

Zero (0) for false and one (1) for true. Any other value is interpreted as false.

Required

No. If omitted, the default value of the element is zero (false) .

Remarks

The heartbeat function of OEP keeps the OEAS session running while the LCD is open. If this element is false, user sessions may expire if the LCD is kept open longer than the session timeout

period for OEAS.

listId Element

The listId element of the XListControl configuration contains the unique identifier for the list configuration.

Legal values

The identifier must be limited to alphanumeric characters. No white space or punctuation characters are allowed.

Required

Yes.

maxRowsMax Element

The maxRowsMax element of the XListControl configuration contains an upper limit to the maximum rows a user may enter into the List Configuration Dialog.

Legal values

Any positive integer.

Required

No. If omitted, the default value is 999.

minMaxRows Element

The minMaxRows element of the XListControl configuration contains a lower limit to the maximum rows a user may enter into the List Configuration Dialog.

Legal values

Any positive integer.

Required

No. If omitted, the default value is 5.

pageView Element

The pageView element of the XListControl configuration identifies the initial view format for the list.

Legal values

Zero (0) for list view and one (1) for page view. Any other value is interpreted as page view.

Required

No. If omitted, the default value is one (page view).

rowIndicatorState Element

The rowIndicatorState element of the XListControl configuration indicates whether the list rows respond to mouseover events by highlighting their contents.

Legal values

This element can contain any of the following integer values:

- 0 - for no highlight
- 1 - for expand icon only highlight
- 2 - for row text and icon highlight

Any other value is interpreted as two (both row and icon highlight).

Required

No. If omitted, the default value is two (both row and icon highlight).

Remarks

When this element is set to enable highlight, text and/or icons in rows in a rendered list change color as the mouse cursor passes over them. On slower computers this effect may cause the list to react sluggishly to user actions. Set this element to zero to disable the highlighting feature.

showRowCount Element

The showRowCount element of the XListControl configuration indicates whether the rendered list will display a count of the total number of rows it contains.

Legal values

Zero (0) for false and one (1) for true. Any other value is interpreted as true.

Required

No. If omitted, the default value is one (true).

Remarks

If the showSelected element is true, this element is ignored.

showSelected Element

The showSelected element of the XListControl configuration indicates whether a counter that displays the total number of selected rows in the list appears.

Legal values

Zero (0) for false and one (1) for true. Any other value is interpreted as true.

Required

No. If omitted, the default value is one (true).

Remarks

The counter appears in the center of the table's bottom border.

dataFields Element

The dataFields element contains one or more child dataField elements. Each child element contains implementation data for the columns used in the list. Each possible child element for dataField appears in the table below.

Element	Description
alternateSortField	Identifies a different data field to use for sorting purposes
bodyImageSource	Identifies a file prefix to use for icons that appear in column cells
captionAuto	Indicates whether to use database caption information for the column heading in the rendered list
captionResId	Identifies a resource string to use for the column heading in the rendered list
cellType	Indicates the type of column to render in the list
cellWrap	Indicates if the column should wrap or truncate long data strings
columnStyle	Identifies the style (link or normal text) for the data in the column
dataFieldId	Contains the unique identifier for the data field
dataSourceColumn	Identifies the column that provides the data displayed in the column
defaultColumnWidth	Indicates the width of the column as a percentage of the total list
eventParameter	Indicates whether the column data should be passed to the event handler
headerImageSource	Identifies a graphic file to use as a column heading for a column that contains icons
initialSortDirection	Identifies the sorting direction for the column when the table is initially rendered
justify	Indicates the position of the data within the column's cells
maxColumnWidth	Indicates a maximum pixel width for the column
minColumnWidth	Indicates a minimum pixel width for the column
sequenceNumber	Identifies the order in which the column appears in the list
sortDataType	Indicates a type to use when sorting data within the column
sortSequence	The order that a column's data should be considered when the list is sorted just before rendering
type	Indicates how the data will be used in the list
userSortable	Indicates whether users can sort the list by the data in the column

alternateSortField Element

The alternateSortField element of the XListControl configuration identifies a different field to use as the criteria for sorting the list.

Legal values

The identifier is limited to alphanumeric characters. No white space or punctuation characters are allowed. This value should contain the name of another column's dataFieldId element.

Required

No. The element has no default value.

Remarks

This element is typically used to allow for sorting by hidden data fields. When the user clicks the column heading to sort the list, the XListControl object uses the data from the linked field as the sorting criteria.

bodyImageSource Element

The bodyImageSource element of the XListControl configuration identifies the prefix for a set of graphic files that can appear in a column's row cells in a rendered list.

Legal values

Any valid XML that identifies a file name prefix for the header icons.

Required

This element is only required if the cellType element has a value of 'icon'. In all other cases the value of this element is ignored.

Remarks

See the topic Using icons in list columns for more information on using this element in a list.

When rendering a list, an XListControl object searches for the file identified in this element in the directory identified in the *IconFilePath* property.

captionAuto Element

The captionAuto element of the XListControl configuration indicates whether the control should automatically generate column headings from the column descriptions provided in the retrieveList data set.

Legal values

Zero (0) for false and one (1) for true. Any other value is interpreted as true.

Required

No. If omitted, the default value is zero (false).

Remarks

The XListControl class can only automatically create column headings from retrieveList data sets. This functionality is not possible with retrieveCollection data sets. The column heading information is

generated as part of the *appendResources* method of the XListData class.

See Auto-generating column headings for more information on using this element.

captionResId Element

The captionResId element of the XListControl configuration identifies a string in a resource file to use for the column heading.

Legal values

The identifier must be limited to alphanumeric characters and the underscore (_) character. No white space or punctuation characters are allowed.

Required

No. The element has no default value.

Remarks

This element refers to the 'name' attribute of a string element within a resource file. See the topic Contents of the resource files for more information about the new resource file format.

If this element is missing or has no value and the column is not automatically configured through the captionAuto element, the column heading in the rendered list will be blank.

cellType Element

The cellType element of the XListControl configuration indicates the type of column to render in the list.

Legal values

The element must contain one of the following values.

Value	Description
boolean	Renders a check for data items that are either true or false (or 1 and 0)
checkbox	Renders a clickable checkbox to allow users to select one or more rows for additional processing (for example, deletions)
expand	Renders an icon that allows the user to view the detail area
icon	Renders an icon in the list cells
Text	Renders the source data as is

Required

Yes.

Remarks

See Using icons in list columns for information about using the boolean and icon values in a list.

cellWrap Element

The cellWrap element of the XListControl configuration indicates whether long data strings in a column should wrap or be truncated.

Legal values

Zero (0) for false and one (1) for true. Any other value is interpreted as true.

Required

No. If omitted, the default value is zero (false).

Remarks

If the element is true, data that is too long to fit in the column is displayed in full by wrapping it in the table cell (and thereby increasing the vertical size of the entire row). If the element is false, long data is truncated after a single line. Users can view the complete data string in a tooltip by placing the mouse pointer over the affected cell.

columnStyle Element

The columnStyle element of the XListControl configuration indicates whether the data should appear in the cell as a clickable link.

Legal values

'normal' and 'link'.

Required

No. If omitted, the default value is 'normal'.

Remarks

If this element is set to 'link', then the data text appears as a clickable link.

When a user clicks the link in the row, the XListControl object generates an event with a cellLink event code. The eventOtherData element of the event XML contains the dataFieldId for the column that was clicked. This is useful if you define multiple columns in a list as containing clickable links.

Columns that can be used as hyperlinks are marked with an asterisk in the List Configuration Dialog.

dataSourceColumn Element

The dataSourceColumn element of the XListControl configuration identifies the source of the data for the column within the list.

Legal values

Column identifiers are comprised of a lower-case letter 'c' and a number (for example, c0).

Required

See below.

Remarks

This element must be supplied for data fields with a cellType of 'boolean' , 'icon', or 'text'. It is ignored if the cellType is 'checkbox' or 'expand'.

defaultColumnWidth Element

The defaultColumnWidth element of the XListControl configuration identifies the starting width of the column expressed in pixels.

Legal values

Any positive integer less than the maxColumnWidth default and greater than the minColumnWidth default.

Required

Only for columns of cellType text.

When the table is rendered, the column width will be this size or the size necessary to show the entire column heading text, whichever is greater.

eventParameter Element

The eventParameter of the XListControl configuration indicates whether the data field's column should be passed to the event handler.

Legal values

Zero (0) for false and one (1) for true. Any other value is interpreted as false.

Required

No. If omitted, the default value for this element is zero (false).

Remarks

Columns with a cellType of checkbox or expand cannot be used as event parameters.

See Event handling for more information on processing parameters in events.

headerImageSource Element

The headerImageSource element of the XListControl configuration identifies a graphic file that can be used as a column heading in the rendered list.

Legal values

Any valid XML that identifies a file path to an image on the Web server.

Required

No. The element has no default value.

Remarks

See the topic Using icons in list columns for more information on using this element in a list.

When rendering a list, an XListControl object searches for the file identified in this element in the directory identified in the *IconFilePath* property.

initialSortDirection Element

The initialSortDirection element of the XListControl configuration identifies the direction a column should be sorted when the list is first rendered.

Legal values

'asc' for ascending sort and 'desc' for descending sort.

Required

No. The element has no default value.

Remarks

If multiple columns have a value for this element, they must also have values for their sortSequence elements.

justify Element

The justify element of the XListControl configuration identifies the position of the data text within the column cells.

Legal values

'left', 'center', and 'right'.

Required

No. If omitted, the default value is 'left'.

maxColumnWidth Element

The maxColumnWidth element of the XListControl configuration identifies the maximum width of the column expressed as a number of pixels.

Legal values

Any positive integer value less than or equal to 2000.

Required

No. If omitted, the element has a default value of 2000.

Remarks

This element is only used for columns with a cellType of text.

This element provides a maximum pixel width for a column within the list. Users will not be able to increase the size of the column beyond this width.

To prevent a user from making changes to the size of a column, set the value of this element and the minColumnWidth element to be identical.

minColumnWidth Element

The minColumnWidth element of the XListControl configuration identifies the minimum width of the column expressed as a number of pixels.

Legal values

Any positive integer value greater than or equal to 85.

Required

No. If omitted, the default value is 85.

Remarks

This element is only used for columns with a cellType of text.

If the caption text for the column heading requires a width greater than the value of this setting, the column is automatically expanded to accommodate the heading text.

This element provides a minimum pixel width for a column within the list. Users will not be able to decrease the size of the column beyond this width (or the width of the column heading text).

To prevent a user from making changes to the size of a column, set the value of this element and the maxColumnWidth element to be identical.

sequenceNumber Element

The sequenceNumber element of the XListControl configuration indicates the position of the data field within its type.

Legal values

Any integer value greater than zero. The first column of a type must have a sequenceNumber of one (1). Values for columns of identical type must be sequential.

Required

See below.

Remarks

This element identifies the sequence for data fields with matching type elements.

For data fields of type 'column' and 'detail', this element is required and dictates the position of the field's column in the rendered table. For data fields of type 'hidden' and 'system', this element is not required and is ignored if provided.

> **Important:** Values for this element must be sequential. Gaps in the value of this element between data fields of identical type will result in empty columns in the rendered list.

sortDataType Element

The sortDataType element of the XListControl configuration identifies how the text for the field should be interpreted for sorting purposes.

Legal values

'date', 'number', or 'text'.

Required

No. If omitted, the default value is 'text'.

Remarks

Tip: If sorting date values in Universal Coordinated Time (UTC) format, it is often more efficient to sort the values as text rather than date.

sortSequence Element

The sortSequence element of the XListControl configuration identifies the sequence in which the data field will be included in the pre-rendering sort of the list data.

Legal values

1, 2, or 3.

Required

No. The element has no default value.

Remarks

This element is only necessary if the configuration has more than one field with its initialSortDirection element set.

If a list is configured to sort by multiple fields when it loads, only the primary sorting column will have a sorting indicator in the column heading.

type Element

The type element of the XListControl configuration indicates how the data field will be used in the list.

Legal values

'column', 'detail', 'hidden', and 'system'.

Required

Yes.

Remarks

The four possible values are interpreted as follows.

Value	Description
column	Appears as a column in the header row of the list.
detail	Appears as a data item in the detail area of the list.
hidden	Does not appear in the list when first rendered, but may be added to the list using the LCD.
system	Does not appear in the list and cannot be added to the list with the LCD. Can still be used as an event parameter.

When the value of this element is 'column' or 'detail', the data field must also have a value for its sequenceNumber element.

userSortable Element

The userSortable element of the XListControl configuration indicates whether users can sort the list based on the values in the data field's column.

Legal values

Zero (0) for false and one (1) for true. Any other value is interpreted as false.

Required

No. If omitted, the default value is zero (false).

Maximum Rows Settings

For performance reasons, the default maximum rows settings for result list control lists are stored in a file at OEP's root directory. The file is called rlc_defaults.xml and contains an entry for every list in OEP that uses the result list classes.

A sample from this file appears below.

```
<globalListConfiguration>
<list id="incidentAuditlogList">
<enableMaxRows>1</enableMaxRows>
<defaultPageSize>300</defaultPageSize>
</list>
...
```

Each list in OEP has a list element with two child elements. The first element, enableMaxRows, indicates whether the user will be allowed to make changes to the maximum rows settings for a list by using the List Configuration Dialog (LCD). Set this element to 1 (one) for true and 0 (zero) for false.

The second element, defaultPageSize, sets the default maximum rows limit for the list. If enableMaxRows is set to true, then users can override this limit by making changes in the LCD.

Note: The result list control does not enforce any maximum rows settings, it just stores a value that can be used by OTMLBOCall objects when making database queries.

If you add a list to OEP, create an entry for it in this file.

Maximum rows for legacy lists

Several OEP features do not use the result list control. Instead, they use lists that have a fixed value for maximum row retrieval, and that value can only be modified through the application code (the exception to this is List Manager, which can be modified by user preferences).

The default setting for these lists is 300 records. To modify this on a global application scale, edit the file YourOEPwebsite/application/onyxapplicationconstants.asp and edit the value assigned to the constant LEGACY_MAX_ROWS.

Customizing Forecast and Quote Line Item Lists

This section describes how to customize the OEP Forecast and Quotes line item lists. You can customize the following functions of the lists:

- Add, remove, and order columns
- Edit column captions
- Apply properties to columns, such as enabling them to be sortable
- Enable secondary currency selections
- Update user-definable fields

Line-item List Customization

The columns that display information about the products that comprise a forecast or quote are defined within the file YourOEPwebsite\ForecastQuote\QuoteMain_ListDef.xml on the OEP server. The columns appear in the same order as they are listed within the XML. Within the file, each “column” element contains the attributes of the column. The following XML fragment is a sample extract from the file:

```
<XML id=”xListDef”>
<listcols>
 <column>
<coltitle><% = OTMStr2SafeHTML(res_List_Title_Col1)
%></coltitle>
<datafield>*delete</datafield>
 <selectable>N</selectable>
  <sortable>N</sortable>
<quotetype>FQ</quotetype>
```

```
</column>
<column>
....
```

The data within the file is processed as an ASP script, so dynamic configurations can be created. The file is processed as part of the QuoteMain.asp file.

Each column has five properties that are set within the XML. The properties and their purpose are described below.

Property description table

Property	Valid Values	Description
coltitle	Any	The heading for the column in the UI. Modify the constants in the file Res\ForecastQuote\QuoteMain_ListDef_ res.asp.
datafield	lineid rowid productid description unitprice quantity disctype discvalue confidence shipdate comments updatedby insertedby *delete - show delete check box *discounted - discounted value for line *discamount - discount amount for line *discvalue - discount value entered *extended - extended value for line *updatedate - formatted update date *insertdate - formatted insert date User Defined Fields (UDFs) use the format quotelineudfx (where x is 1 -10) The following fields are only used if EnableSecondaryCurrency is set to Yes in the SystemParameters table.	The name of the quoteline field. A prefix of "*" indicates a field that is either calculated or has special formatting or function. UDF datafields can be of the following type: Text box Drop-down Hierarchical drop-down Check box Mask UDF data values correspond to the Reference Administration Tool QuoteLineItem.UserX fields.

Property	Valid Values	Description
	unitprice2 discamount2 discounted2 extended2	
selectable	Y, N	Y - Enables this field to be selected as a link. N - This field is not selectable as a link.
sortable	N, NUMBER, DATE, or STRING	N or the type of the sort. N defines the column as not sortable. The type defines the sort to enable.
quotetype	FQ, F, or Q.	FQ - Configures the column element to appear in both the Forecast and Quote detail interfaces. F - Appear only in the Forecast detail Q - Appear only in the Quote detail.

Note: To change the column caption on the list view, you must modify the resource file Res\ForecastQuote\QuoteMain_ListDef_Res.asp. The default user-defined field captions set through the OEAS Reference Table Administration tool do not appear in this main quote interface (but they do appear in the line-item detail window). See the Customizing resource files topic for more information about resource files.

Other Customization Options

Other forecast and quote customization options are identified here.

Default and secondary currency codes

The default currency code and the secondary currency code can be configured through OES System Parameter Administration. The parameter for the default currency is DefaultCurrencyCode. For this setting, enter the currency that all forecasts and quotes use as their default currency. The parameter for secondary currency is EnableSecondaryCurrency. For this setting, enter Y to enable users to choose an additional currency in which the forecast or quote is calculated.

Change user-defined field entries

Use OES Reference Table Administration to change user-defined field (UDF) captions. OEP only displays UDFs in the user interface if their captions have been edited and an edit has been made to an ASP file on the server. Refer to your OEAS documentation for information on using the Table Administration tool.

To enable the display of UDFs in a quote document, edit the following line in YourOEPwebsite/ForecastQuote/QuoteMain.asp:

```
const DISPLAY_LIST_UDF = false  ' turn UDF display on/off
```

Change the value of the variable to true.

Once this line has been changed, edit the file YourOEPwebsite/ForecastQuote/QuoteMain_ListDef.xml as described in the topic Line-item list customization to enable the display of UDFs within the line-tem lists.

Customizing Quote Templates

You can customize the quote templates to change the appearance of Onyx quote documents or to change the information that is retrieved from the database. The following template files are installed during setup into the YourOEPwebsite\CustomLetter\Templates directory. Two of the templates are designed for use with Microsoft Word, and can be updated by users after generation from OEP.

File Name	Output Format	Description
printquote.asp	Microsoft Word is launched within Internet Explorer.	Use this file for quote documents if Microsoft Word is installed on users' computers.
printquotemc.asp	Microsoft Word is launched within Internet Explorer.	Use this file for quote documents if Microsoft Word is installed on users' computers. This template is designed to support multiple currencies.
printquote_htm.asp	HTML	Use this file for quote documents if Microsoft Word is not installed on users' computers.
printquotemc_htm.asp	HTML	Use this file for quote documents if Microsoft Word is not installed on users' computers. This template is designed to support multiple currencies.

The templates are ASP files that generate HTML from the retrieved quote business objects. The other ASP files in the same folder map the quote object properties to the values in the quote documents.

Customizing Quote Document Appearance

You can modify the HTML code within the quote templates to change the appearance of the quote documents. To change only the appearance of the documents, modify only the HTML code that appears between the <HTML> and </HTML> tags.

Setting output mode

OEP uses the quote document filename to determine the destination for the output data. If the filename ends with "_htm.asp", OEP opens the quote ASP file in a new browser window for display as HTML. If the filename has any other ending to it, OEP assumes the output data is intended for a different application and redirects the ASP file output to a hidden IFRAME. The quote document is then responsible for setting its response headers to identify the destination application. The following VBScript function call appears within OEP's MS Word templates to change the content type of the response to application/msword, which instructs the browser to launch Word to accept the data.

```
call SendOutputToWord()
```

The code for this function is located in the file YourOEPwebsite\CustomLetter\Templates\otm_templateheader.asp. Review the code to adapt the function for other applications that can process custom quote documents.

Generating the table of quote line items

The functions vbWriteQuoteHeader and vbWriteQuoteLines generate quote detail information. Use these functions to create the table that lists the items that comprise the quote. You can duplicate or adapt these functions as necessary to create custom configurations.

Function	Called From	Example
vbWriteQuoteHeader	Within <TR></TR> tags.	<tr id=”Row0” class=”row0”> <%=vbWriteQuoteHeader ()%> </tr>
vbWriteQuoteLines	Within <TABLE></TABLE> tags, but outside of any <TR></TR> tags, as a new table row is generated for each quote line item.	<table border=”0” cellspacing=”1” cellpadding=”1” width=100%> <tr id=”Row0” class=”row0”> <%=vbWriteQuoteHeader ()%> </tr> <%=vbWriteQuoteLines ()%> </table>

Customizing Data Fields in Quote Documents

Quote templates have two types of fields available for configuration, customer fields and quote fields. When a quote document is generated, these fields are populated from the customer and quote business objects. If no value exists, the field contains an empty string.

To reference a field in the template, use the format:

```
<%=[field name]%>
```

For example, if the field you want to reference is fldLastName, type the following:

```
<%=fldLastName%>
```

Alternatively, you can use the following format to reference a field:

```
<%=WriteFieldandBreak([field name])%>
```

For example:

```
<%=WriteFieldandBreak(fldFullName)%>
```

If the field contains a value, the value is populated, and a
 tag is added. If the field does not contain a value, it remains blank and prints nothing. This is useful for object properties that may or may not contain values, as it prevents empty lines from being added to the document.

Customer Reference Fields

Customer and address fields are mapped within the script file YourOEPwebsite\CustomLetter\Templates\otm_GetCustomerDetails.asp. Edit this file to change the mapping between those OEAS business objects and the quote document data. Not all of the business object properties are mapped by default. Those that are appear in the following table.

Field Name	Field Name	Field Name
fldPrefix	fldDepartmentDesc	fldFormalName
fldFirstName	fldeMailAddress	fldCityStateZip
fldMiddleName	fldTitleCode	fldFullName
fldLastName	fldDepartmentCode	fldPhoneNumber
fldSuffix	fldAddress	fldFaxNumber
fldCompanyName	fldTitleDesc	fldPostCode
fldData1 ... fldData10	fldUser1 ... fldUser10	

Quote Reference Fields

Quote fields are mapped within the script file YourOEPwebsite\CustomLetter\Templates\otm_GetQuoteDetails.asp. Edit this file to change the mapping between the OEAS business object and the quote document.

Field Name	Field Name	Field Name
fldQuoteId	fldQuoteUser1... fldQuoteUser10	fldQuoteConversionFactor
fldQuoteTitle	fldQuoteCurrencyLocale1	fldQuoteTotal1
fldQuoteStatus	fldQuoteCurrencyLocale2	fldQuoteTotal2
fldQuoteExpiryDate	fldQuoteCurrencyCode1	fldQuoteDiscount1
fldQuoteComments	fldQuoteCurrencyCode2	fldQuoteDiscount2
fldQuoteCloseProbability	fldQuoteCurrencyDesc1	fldConfidenceTotal1
fldQuoteCreateDate	fldQuoteCurrencyDesc2	fldConfidenceTotal2
fldQuoteCloseDate		

Configuring Quote Details

The arrQuoteDetailColumns array in the quote template ASP files contains the data fields that appear in the quote lines table of the quote document. You can configure the columns that appear, as well as their width, alignment, and heading text by modifying the contents of the array. The code in the ASP file then automatically generates and formats the document according to the provided settings. The array has two dimensions. The first dimension contains the column number, and the second dimension contains the data and formatting settings for the column. The size of the first dimension is equal to the number of columns in the quote template. The second dimension always has a size of four.

The line of code that immediately precedes the array settings sets the number of columns that appear in the document. Edit the value of the first dimension to change the number of fields in the array. Failing to set this correctly will cause errors when the document is created. The following line sets the array for 8 columns of data.

```
ReDim arrQuoteDetailColumns(8,4)
```

Configuring each column

Each of the four elements of the second dimension of the array has a specific purpose. There are constants for each column that can be used to correctly prepare the data within the array. Each element is described below.

QUOTEDETAIL_FIELD

Use QUOTEDETAIL_FIELD to identify the data field that appears in the column. The fields are mapped from the quote object XML data in the file OnyxWebSite/forecastquote/otm_quote_DAC_inc.asp. See the function vbGetQuoteLinesXML in this file to review and/or update the data field mappings.

QUOTEDETAIL_LABEL

Use QUOTEDETAIL_LABEL to specify the label that appears in the column header. This item can contain any valid HTML string.

QUOTEDETAIL_WIDTH

Use QUOTEDETAIL_WIDTH to specify the width of the column in the table. This item must contain a valid setting for the WIDTH attribute of a table cell (<TD>) element.

QUOTEDETAIL_ALIGN

Use QUOTEDETAIL_ALIGN to specify the alignment of the cell contents for the data in the column. This item must contain a valid setting for the ALIGN attribute of a table cell (<TD>) element.

Examples

The following code fragments show sample settings for values within the arrQuoteDetailColumns array.

```
arrQuoteDetailColumns(5,QUOTEDETAIL_FIELD) = "unitdiscount"
arrQuoteDetailColumns(5,QUOTEDETAIL_LABEL) = "Unit Discount"
```

```
arrQuoteDetailColumns(5,QUOTEDETAIL_WIDTH) = "20%"
arrQuoteDetailColumns(5,QUOTEDETAIL_ALIGN) = "Right"


arrQuoteDetailColumns(6,QUOTEDETAIL_FIELD) = "discountedprice"
arrQuoteDetailColumns(6,QUOTEDETAIL_LABEL) =
"Discounted<BR>Price"
arrQuoteDetailColumns(6,QUOTEDETAIL_WIDTH) = "Auto"
arrQuoteDetailColumns(6,QUOTEDETAIL_ALIGN) = "Center"
```

Creating and Activating Quote Templates

If you want to create a new quote template, start with an existing template and modify it to suit your needs. Save the templates to the YourOEPwebsite\CustomLetter\Templates directory with a file extension of .asp. Remember to use the suffix _htm.asp to instruct OEP to output the quote in a new browser window.

Templates originally used within OEP - Windows Client cannot be used by the Web client. These templates must be converted to ASP files. Currently, there are no tools to automate this process.

To add or remove quote templates from use within the OEP user interface, use the OES Table Maintenance Administration Tool to edit the details of each template within the Template table. Review the Onyx Enterprise Studio documentation for information on using Table Maintenance.

Customizing List Manager

You can add use OES Table Administration to add database views to List Manager.

After you've added the desired views, you can modify their appearance and usability by customizing several different files on the OEP server.

Customizing Result Set Click Actions

The data columns that appear in result sets of List Manager queries can be mapped to specific browser script functions through an XML file located on the OEP server. This mapping then allows a user to click different parts of a result set row to open other OEP records. The configuration file contains information about the actions that occur when a user clicks one of the rows in the result set table.

The file is YourOEPwebsite/listmanager/lm_resultType_map.xml. An excerpt from the file appears below.

```
<actionDefinition>
 <platform name="oracle">
<columns objectType="1"
```

```
bulkActionColumnType="individualPrimaryId">
 <column columnType="individualSecondaryId">
  <onClick event="vbCustomerClick" type="script">
  <param columnType="individualPrimaryId"/>
   </onClick>
 </column>
 ...
```

Note: There is one valid <platform> in this release of OEP: oracle. The oracle name attribute indicates that the OEAS platform is using generated SQL, or SQLGen, to access the database. Ignore the <platform name="mssql"> node.

The <columns> element contains <column> elements for all clickable items associated with a view (the view is identified by the objectType attribute). In the default configuration, only one column in each view (companies and individuals) is configured with an action: secondaryId.

By altering the configuration you can change the resulting action depending on where the user clicks in each row. The supported elements in the configuration file appear in the following table.

Element	Description
platform	Identifies the underlying database access platform
columns	Contains the list of columns that can be clicked to execute a script function
column	Identifies a clickable column and contains elements and attributes to describe the click action
onClick	Identifies the function called when the item is clicked and contains the parameters to pass to the function
param	Identifies a parameter to pass to the function

The settings for each of the above elements exist in the form of XML attributes. The supported attributes (and the elements they can be used with) appear in the following table.

Attribute	Elements	Description
bulkActionColumnType	columns	Identifies the column used as the key for bulk actions for the selected objectType.
columnType	column, param	For the column element, this property identifies the column that can be clicked. For the param element, this identifies the column whose data is passed to the action script function.
event	onClick	Identifies the function called when the column is clicked.

Attribute	Elements	Description
name	platform	Identifies the platform the configuration is used with.
objectType	columns	Identifies the list manager view by database object type (companies or individuals).
type	onClick	Identifies the source of the action code associated with the onClick event. The only current type is script.

The source values for the columnType and bulkActionColumnType attributes are configurable within the Table Administration tool of the OES. To add a new value, use the tool to edit the result_type column of the search_view_detail table. Review the Onyx Enterprise Studio documentation for information on using the Table Maintenance administration tool.

The arguments for the called function are passed in the order the <param> elements appear as children of the <onclick> element. The functions for a custom action should be included in the file YourOEPwebsite/listmanager/lm_results.asp.

Limiting the Size of Result Sets

Since large list manager searches can potentially create performance issues on a production server, there is a system setting to limit the values a user can enter into their list manager preference settings. This limitation is checked each time a user edits their General Preference settings.

The value is defined by the constant PREFERENCE_LM_MAXROWS_LIMIT located within the file YourOEPwebsite/defaults/onyxdefaults.inc. This constant is used to validate the only the value entered into the preference form. It does not set the default value for the user preference.

Setting the value of the constant to zero (0) disables the checking in the preferences form.

Configuring Bulk Actions

List Manager result sets are often generated for the purpose of processing multiple OEP records at a single time. The availability of these bulk actions (sometimes also referred to as customer processes), is configurable through an XML file located on the OEP server.

The file is named lm_bulkaction.xml and is located at within the YourOEPwebsite/listmanager folder. An excerpt from the file appears below:

```
<bulkActions>
 <bulkAction id="exportExcel">
  <views>
   <view>all</view>
  </views>
  <platforms>
 <platform>all</platform>
```

```
</platforms>
 </bulkAction>
...
```

Each bulk action is represented by a bulkAction element within the XML. The id attribute identifies the bulk action to configure. These identifiers are coded into the list manager feature and should not be changed.

Each bulkAction element can have two children, views and platforms. The views element identifies the views with which the bulk action is usable. Views can be identified by number (from the value of the primaryId column where the view is defined in OES Table Administration), or all views can be associated with the action by creating one child view element that contains the word 'all' (as shown above).

The platforms element contains the names of the platforms with which the bulk action is usable. The valid values are 'oracle' and 'all'. The 'mssql' value is not valid for this release of OEP. Note that these values are case-sensitive.

The actions appear in a drop-down list in the list manager interface in the order they are listed in the XML file. Actions do not appear in the list if they do not have the proper permissions identified within the file. To remove an action from the UI completely, do not assign it any views or platforms (or remove it from the file).

The following example configures the 'Create a literature order for selected items' bulk action by limiting it to the Individuals view.

```
<bulkAction id="literature">
 <views>
<view>1</view>
 </views>
<platforms>
 <platform>oracle</platform>
 </platforms>
</bulkAction>
```

Opening OEP Records from Other Applications

OEP provides a mechanism from which non-browser applications can call OEP to retrieve database records or open blank forms for data entry. There are two files on the Web server that can be used as shortcut entry points to some of the edit windows within OEP. The shortcuts can also be used to perform simple record searches. To learn more about using these shortcuts, click a topic in the list below.

Access Methods

An external application can access OEP by opening one of two files located on the Web server. Each shortcut file represents a different method for accessing OEP. The first, known as the direct method, is for applications that can create an instance of a browser window with a name property that can be referenced using standard DHTML script tools. Generally this requires that the calling application is either already running within an instance of Internet Explorer, or is able to create one through OLE automation (for example, from within a Microsoft Office application using Visual Basic for Applications, or from any COM-enabled Win32 application).

The second method, known as the indirect method, is for applications that use the Windows Explorer shell to open a browser through a file association of type URL. A common example of this is clicking a URL from within an email opened in a third party application. The indirect method is less desirable because it requires OEP to open a temporary window which then calls the direct method. This temporary window remains open and is effectively abandoned by OEP when the user begins working with the new window. The user must manually close the temporary window to remove it from the desktop.

> **Important:** Shortcut actions require that an active OEP session exist. If one does not, the user is required to log in. The user must then re-execute the shortcut action.

The indirect method may also inadvertently cause an OEP user to lose an existing session. If the Internet Explorer configuration option **Reuse windows for launching shortcuts** is set, an indirect link may cause an open OEP browser window to surrender itself to the opening link.

The action of the shortcut is determined by the items in the query string portion of the URL. Each method uses a different file to open the shortcut, but both interpret the query string data identically. A shortcut link can do one of three things:

- Open a blank customer, incident, work ticket, or script form for the user to input data
- Open an existing customer, incident, work ticket, script, or email record for the user to review and/or edit
- Display the results of a customer, incident, or work ticket search (either a list of records or a single record, depending on how many records match the search criteria)

The two shortcut files are *YourOEPwebsite*\powerpage\oep_launcher.asp (for indirect access) and *YourOEPwebsite*\powerpage\oep_automate.asp (for direct access). See Query-string parameters for information on constructing a query string.

Query-string Parameters

The query string must identify the action (new, retrieve, or search) and the type of record involved (customer, incident, work ticket, or script). The other necessary items in the query string are dependent on the first two. The following table lists all possible items that can be added to a query string. For example strings, see Sample query strings.

Parameter	Description
Action	Identifies what the shortcut should do (open a blank form, retrieve an existing record, or perform a search)
Type	Identifies what type of record is involved (customer, incident, work ticket, or script)
Id	Contains the unique identifier of a record to retrieve
IdType	Identifies the type of ID to be used in the search (primary or secondary)
Category	Identifies the type of customer or incident to insert (individual, company, sales, service, or support)
OwnerId	Contains the unique identifier of the customer record that will own the newly created incident
OwnerIdType	Identifies the type of ID that the OwnerId parameter represents (primary or secondary)
FilterField	Identifies the data field to be used in a search
FilterValue	Contains the value of the data field to match in the search
ScriptId	Identifies a script to execute
ScriptMode	Indicates whether to execute a specific script or to display the script chooser
OwnerType	Indicates the type of record (customer or incident) that will own the new script

Characters in the query string must be part of the standard ASCII character set. Unicode and other multi-byte characters are not supported.

To keep the size of the query string to a minimum, the Action, Type, Category, OwnerIdType, IdType, FilterField, and ScriptMode parameters use numeric constants to identify the options they represent. These constants are declared in the file oep_automate.asp and can be copied to other applications that will be creating shortcut URLs. The acceptable values for each parameter appear in the following tables.

Valid values for the Action parameter are:

Parameter constant	Value	Description
OEP_AUTOMATION_ACTION_NEW	0	Opens a new record form
OEP_AUTOMATION_ACTION_LOAD	1	Loads an existing record
OEP_AUTOMATION_ACTION_SEARCH	2	Executes a search

Valid values for the Type parameter are:

Parameter constant	Value	Description
OEP_AUTOMATION_TYPE_CUSTOMER	0	Customer
OEP_AUTOMATION_TYPE_INCIDENT	1	Incident

Parameter constant	Value	Description
OEP_AUTOMATION_TYPE_WORKTICKET	2	Work ticket
OEP_AUTOMATION_TYPE_SCRIPT	3	Script
OEP_AUTOMATION_TYPE_EMAIL	4	Email

Valid values for the Category parameter are:

Parameter constant	Value	Description
OEP_AUTOMATION_CATEGORY_INDIVIDUAL	0	Individual
OEP_AUTOMATION_CATEGORY_COMPANY	1	Company
OEP_AUTOMATION_CATEGORY_SALES	3	Sales incident
OEP_AUTOMATION_CATEGORY_SERVICE	2	Service incident
OEP_AUTOMATION_CATEGORY_SUPPORT	1	Support incident

Valid values for the IdType parameter are:

Parameter constant	Value	Description
OEP_AUTOMATION_TYPE_PRIMARY_ID	0	Primary ID
OEP_AUTOMATION_TYPE_SECONDARY_ID	1	Secondary ID

Valid values for the OwnerIdType parameter are:

Parameter constant	Value	Description
OEP_AUTOMATION_OWNERTYPE_PRIMARY_ID	0	Primary ID
OEP_AUTOMATION_OWNERTYPE_SECONDARY_ID	1	Secondary ID

Valid values for the FilterField parameter are:

Parameter constant	Value	Description
OEP_AUTOMATION_SEARCH_CUSTOMER_ SECONDARY_ID	0	Searches on the unique ID for customer records
OEP_AUTOMATION_SEARCH_CUSTOMER_ FIRST_NAME	1	Searches on the first name for customer records
OEP_AUTOMATION_SEARCH_CUSTOMER_ LAST_NAME	2	Searches on the last name for customer records
OEP_AUTOMATION_SEARCH_CUSTOMER_ COMPANY_NAME	3	Searches on the company name for customer records
OEP_AUTOMATION_SEARCH_CUSTOMER_ POST	4	Searches on the postal code for customer records

Parameter constant	Value	Description
OEP_AUTOMATION_SEARCH_CUSTOMER_PHONE	5	Searches on telephone numbers for customer records
OEP_AUTOMATION_SEARCH_INCIDENT_SECONDARY_ID	0	Searches on the unique ID for incident records
OEP_AUTOMATION_SEARCH_INCIDENT_OWNER_ID	1	Searches on the owner ID for incident records
OEP_AUTOMATION_SEARCH_INCIDENT_DESCRIPTION	2	Searches on the descriptions for incident records
OEP_AUTOMATION_SEARCH_WORKTICKET_SECONDARY_ID	0	Searches on the unique ID for work ticket records
OEP_AUTOMATION_SEARCH_WORKTICKET_DESCRIPTION	1	Searches on the description for work ticket records

Valid values for the ScriptMode parameter are:

ParamUeter constant	Value	Description
OEP_AUTOMATION_SCRIPT_INSERT_MODE_CHOOSER	0	Opens the script chooser window
OEP_AUTOMATION_SCRIPT_INSERT_MODE_DELIVERY	1	Opens a new or existing script by unique ID

Valid values for the OwnerType parameter are:

Parameter constant	Value	Description
OEP_AUTOMATION_TYPE_CUSTOMER	0	Assigns the script to a customer
OEP_AUTOMATION_TYPE_INCIDENT	1	Assigns the script to an incident

Sample Query Strings

The following URL samples show how to execute shortcut actions.

Inserting new records

To open a new work ticket edit form using a shortcut, the query string need only include the action and the type:

```
http://servername/YourOEPwebsite/powerpage/oep_automate.asp?Action=0&Type=2
```

To open a new customer edit form, the query string must also include the customer category (company or individual):

```
http://servername/YourOEPwebsite/powerpage/oep_
```

launcher.asp?Action=0&Type=0&Category=0

To open a new incident edit form, the query string must also include the owner customer ID, the owner ID type, and the incident category:

http://servername/YourOEPwebsite/powerpage/oep_automate.asp?Action=0&Type=1&OwnerId=38D178C1-9CA9-4E4B-B169-ED022AB2C539&OwnerIdType=0&Category=3

Retrieving existing records

To retrieve a work ticket, incident, email, or customer record using a shortcut, the query string must contain the action, the type, the ID of the record, and the type of the ID (primary or secondary):

http://servername/YourOEPwebsite/powerpage/oep_launcher.asp?Action=1&Type=2&id=45913&IdType=0

Executing searches

The query strings for searches must contain the type of record to search for, and a single criterion to be used in the search. The available criteria are listed in oep_automate.asp (and are identical to some of those available from the quick search feature). For example, the following URL searches for incidents by description:

http://servername/YourOEPwebsite/powerpage/oep_automate.asp?Action=2&Type=1&FilterField=2&FilterValue=system%20problem

Note: Depending on your network configuration, URL escaping rules may apply. Filter criteria with space characters should be properly escaped to prevent errors.

Executing scripts

The query strings for scripts can do one of three things: load an existing script for review, start a new script, or open the script chooser window.

To load an existing script for review, the query string must contain the action, the record type, the script ID, and the type of the script ID (primary or secondary):

http://servername/YourOEPwebsite/powerpage/oep_automate.asp?Action=1&Type=3&id=2398&IdType=0

To start a new script session with a specific script, the query string must contain the action, the type, the script mode, and the script ID. The owner ID and owner type values are optional. If they are not included, the user must manually assign the script to an owner before it can be saved.

http://servername/YourOEPwebsite/powerpage/oep_launcher.asp?Action=0&Type=3&OwnerId=EFE60935-5F7C-43B0-A597-8D6933805786&OwnerType=1&ScriptMode=1&ScriptId=763

To open the script chooser dialog, the query string must contain the action, the type, and the script mode. The owner ID and owner type values are optional. If they are not included, the user must manually assign the script to an owner before it can be saved.

> http://servername/YourOEPwebsite/powerpage/oep_launcher.asp?Action=0&Type=3&OwnerId=DEA1BE39-8E09-4979-8BF4-50CCE7EE3CD5&OwnerType=1&ScriptMode=0

OEP Customization Examples

This book provides several examples of how you can customize OEP on your own, including:

Group	Description
Toolbar button	Demonstrates how to add a custom toolbar button.
Masked edit	Demonstrates how to provide masking in client-side environment (using the masked edit control) and server-side environment (using the masked data object).
OTMHelper	Demonstrates how the use the various OTMHelper classes to retrieve, insert, and update customer information via the Onyx Transaction Manager (OTM).
Security	Demonstrates how to secure custom ASP pages within the OEP architecture.
Result list control	Demonstrates how to create a list using data retrieved from the database.

File locations

The following customization examples frequently refer to files or subdirectories that are located in the OEP site folder. This section refers to the root directory of the OEP installation directory as 'YourOEPwebsite'. For example, if you installed OEP in the directory 'c:\inetpub\wwwroot\OnyxEmployeePortal', that directory is referenced in this documentation as YourOEPwebsite. If the reference in the documentation is to the directory 'YourOEPwebsite\StyleSheet', the actual directory on your system is 'c:\inetpub\wwwroot\OnyxEmployeePortal\StyleSheet'.

The files required to run these customization examples are located in the following directories on the OEP product CD:

Files	Location
OEP website	Support\WebSite
Customization Guide ASP examples	Documentation\Developer's Reference\Examples

Note: The customization examples must be saved to a subdirectory named Examples within the directory that contains the OEP website. For installation instructions, see the Readme.txt file in the Examples directory.

Customization environment

For many of the examples, Microsoft Visual Studio is the recommended tool for development and testing. With Visual Studio, you can create a project for OEP customization and testing, import the scripts from the OEP Web site, then edit and debug your customizations.

Coding conventions

Onyx recommends prefixing VBScript functions with a 'vb' designator, and JavaScript functions with 'js'.

As a general rule, Onyx has no preference for a primary language for client-side scripting. The only notable exception is that JavaScript cannot pass parameters by reference. For example, if your script needs to pass parameters to the masked edit control, use VBScript in those cases.

Adding a Custom Toolbar Button

This customization shows how to add a button to the toolbar on the Survey Details page. The code within the custom files assumes that the company business object has been modified to store stock ticker symbol data in the property user5. If your implementation stores this data in a different property, modify the sample file YourOEPwebsite/Examples/Toolbar/customaction.js.

Note: For details on adding a custom toolbar button to a UCW-enabled page, see Adding toolbar buttons and other items.

Modifying surveys_details.asp

Toolbar buttons on the PowerPage are created by the code within *YourOEPwebsite*/powerpage/surveys_details.asp. The icons are arranged within a table and stored within IMG elements with JavaScript functions attached to various events. The code also checks for specific permissions before adding some of the buttons to the toolbar. If the user does not have sufficient permissions, the button is omitted from the output HTML.

There are three places in survey_details.asp that must be modified for this example.

The first is near the beginning of the file where the server-side ASP files are included. This code includes the script file that contains the function to add the button to the toolbar.

```
...
<!-- #INCLUDE FILE="../common/include/CSPathtoRoot.asp" -->
<!-- CSBegin custom button include file -->
<!-- #include file="../Examples/Toolbar/drawcustombutton.asp" --
>
```

```
<!-- CSEnd custom button include file -->

<!-- #INCLUDE FILE="../application/onyxapplicationconstants.asp"
-->

...
```

The second change is just below the first where the client-side scripting files are included. These files contain the functions that perform the actions of the customization. The code is broken into two parts, the main function and a resource file.

```
...

<script language="javascript"
src="../onyxcommon/common/javascript/xmlcommon.js"></script>

<!-- CSBegin custom button script files -->

<script language="javascript"
src="../Examples/Toolbar/customaction_res.js"></script>

<script language="javascript"
src="../Examples/Toolbar/customaction.js"></script>

<!-- CSEnd custom button script files -->

...
```

The final change is near the bottom of the file and just below the HTML that generates the refresh button. This is a function call to the code within drawcustombutton.asp. This is the function that creates the IMG element for the button. The function checks the current customer type. If the customer is an individual, the function does not output HTML and the button does not appear on the toolbar.

```
...

<!-- CSBegin custom button rendering -->

<img id="iconCustom" src="../images/icons/iconpapclip_
clear0.gif" onClick="jsCustomAction()" alt="Custom Action"
enableThis = "true" <%vbWriteImageEvents(true)%>> 

<!-- CSEnd custom button rendering -->

<IMG id="idClearIcon" src="../images/icons/iconerase0.gif"
onclick="javascript:jsClearSurvey()" alt="<%=res_Clear_Survey%>"
enableThis = "true" <%vbWriteImageEvents(true)%>> 

...
```

After the files are in place, you should be able to test the customization by opening a company record on the PowerPage. For information about the underlying code, see the next topic, How the customization works.

How the Customization Works

After the files are in place you can test the sample by loading a company record and clicking  in the toolbar. The JavaScript function checks the cached customer business object XML for a value in the property user5. The value of the property is then added to a URL for the CNN financial Web site, which opens in a new browser window. If no data is present in the property, a message box appears that explains that no stock information is available for the customer.

Adding the button

The function vbDrawCustomButton in the file drawcustombutton.asp checks the type of the customer being loaded onto the PowerPage. If the type is company, the button is added to the output HTML for customerheader.asp. The code that adds the button appears in the following code fragment.

```
' If the customer type is company, draw the button.
if lcase(strCustType) = "company" then
' Button tag
Response.Write "<img align='middle' id='iconCustom'
name='iconCustom' _
src='../images/icons/iconpapclip_clear0.gif'
onClick='jsCustomAction()' alt='Custom Action' "
' This function writes out the mouseover and click actions.
vbWriteImageEvents(true)
' And close the img tag.
Response.Write ">"
else
... ' Do nothing
```

The example uses a button graphic 'borrowed' from the Messenger feature (a paperclip with a line through it). Note the use of the vbWriteImageEvents function. This function outputs the standard set of events with functions that automatically handle mouseover and click actions. For information about the buttons in use throughout OEP, see Button standards.

Opening the window

When a user clicks the custom button, the function jsCustomAction in customactions.js does two things:

- It obtains the stock ticker symbol data from the cached business object XML.
- Displays the information for the user.

The code that locates the stock symbol data appears in the following code fragment.

```
...
```

```
strType = top.goCurrentCustomer.getCustomerType();
// If we have a company (which we always should)
// look for the stock ticker symbol.
if (strType == '2')
{
// Find the element for the stock ticker from the local DOM.
try {
oNode = top.goCurrentCustomer.getCustomerXmlDom
().selectSingleNode("//company/user5");
}
...
```

The object goCurrentCustomer is a global object that is available in the main frame's scope. The getCustomerType method retrieves customer type value by reading the cached XML. The getCustomerXmlDom method obtains a reference to the cached XML for the customer record. From here you can collect information from the XML using the standard set of DOM functions.

The function retrieves the data from the object property and verifies that the string is not empty. If so, it opens a new window that shows the information for the company's stock price. If there is no data in the object property, a message box appears that informs the user that no stock information is available for the customer.

This customization could be modified to retrieve any data from the company or individual business object. By creating custom properties through the OEAS you can add just about anything to a business object and use it within OEP.

Button Standards

All buttons in OEP follow a simple standard for implementation purposes. By following this standard you can add buttons whose mouse events are handled by OEP common code.

Tip: The UI Configuration Workbench (UCW) lets you configure toolbars and their buttons on certain OEP pages. See Configuring OEP and About UCW for more information. Toolbar buttons added with UCW conform to the standards listed here.

Button states

Each button has four states. Each state is described in the following table.

State	Description	Example
0	Normal - transparent background, no border.	
2	Mouseover - colored background, colored border, contents shifted one pixel up and one pixel to the left, contents have a grey shadow.	

State	Description	Example
3	Clicked (mousedown) - colored background (darker than state 2), colored border, contents in their normal position (not shifted).	
4	Disabled - transparent background, icon contents grey.	

All Onyx icons are 23 pixels wide by 22 pixels tall and have a color depth of 8 bits. The number for the state of the icon is appended to the name before the extension. The icons in the above table are named iconQSMerge0.gif, iconQSMerge2.gif, iconQSMerge3.gif, and iconQSMerge4.gif.

Masked Edit Examples

The masked edit examples demonstrate the use of the client-side masked edit control, and the server-side masked data object.

Example	Description
Masked edit control	Demonstrates the use of the masked edit control in DHTML scripting on the browser client
Masked data object	Demonstrates the use of the masked data object in ASP scripting

OTMHelper Examples

These examples demonstrate the OTMHelper class usage and are derived from the OEP ASP scripts. The code in the examples has been edited to illustrate the essentials of OTMHelper operation. In practice, you would most likely take advantage of existing functions and subroutines for performing the activities that these examples demonstrate. However, the examples provide you with simple demonstrations without the complexity required of a full application. The files for the OTMHelper demonstrations are located in Examples\OTMHelper\ and Examples\Include.

The OTMHelper examples demonstrate the following features:

Example	Description
Business object methods	Incidents are used to demonstrate the basics for retrieval, creation, updating, and deleting of simple objects.
Multiple business objects	A customer object that has phone and address lists associated with it.
Client-side OTMHelper	The OTMHelper classes can be used in client-side DHTML programming as demonstrated in this example.

Business Object Methods

The business object methods examples demonstrate operations on OEAS business objects using the incidents business object. Incident business objects are simple objects and, as such, are easy to manipulate.

The examples in this section demonstrate the following logical operations.

Method	Description
Insert	Demonstrates how to create and insert a new business object in the OEDB.
Retrieve	Demonstrates how to retrieve and view an existing business object from the OEDB.
Update	Demonstrates how to perform a partial update to an existing business object.
Delete	Demonstrates how to delete an existing business object from the OEDB.

Incident Insert Example

This example demonstrates how to create a new business object and insert it into the OEDB.

The essential elements of an incident insertion are:

- Collect the required input information for the new incident.
- Create an OTMLBOCall object for inserting the new incident object.
- Create an OTMObject parameter and populate it with the information for the new incident object.
- Call the OTM to insert the new incident object.
- Retrieve and display the ID number for the newly created incident object.

Incident Retrieve Examples

The incident retrieve examples demonstrate retrieving a specific incident object, or a set of incident objects owned by a customer.

Single incident retrieve

The essential elements of a single incident retrieval are:

1. The user types an ID and selects Incident as the ID type to retrieve.
2. The user clicks the Submit button sending the form data to the server.
3. The CG_OTMGetIncident function retrieves the information for the specified incident object.
4. The incident object details are formatted in ASP script and sent to the client browser.
5. There are Update and Delete buttons for the user to test these actions.

Customer incidents retrieve

The essential elements of an incident set retrieval are:

1. The user types an ID and selects Customer as the ID type to retrieve.
2. The user clicks the Submit button sending the form data to the server.
3. The CG_OTMGetIncidentByCustomer function retrieves Sales, Service, and Support incidents. Each group of incidents is contained in a rowset.
4. The incident object IDs are formatted in ASP script and displayed in an HTML table. These IDs are linked back to the incident retrieve ASP page so that the user can click any ID to view its details.

Incident Update Example

The incident update example demonstrates how to modify an incident object, and how to partially update to a business object.

The key programming features that this example demonstrates are:

- How to modify properties of a business object.
- How concurrency control is used within OEP and OEAS.

The essential elements of a incident update are:

1. For a given incident object, the description1 and description2 data is presented in two edit windows on a form.
2. The data can be modified and then the Update button clicked to submit the form.
3. The ASP script calls the function CG_OTMUpdateIncident that:
    a. Retrieves the original incident object.
    b. Creates a new OTMLBOCall object for updates.
    c. Calls the bInitializeMerge method to merge the existing incident object into the update OTMLBOCall object.
    d. Applies the modified description1 and description2 data to the update OTMLBOCall object data.
    e. Calls the bExecuteLocal method to send the updated incident object to the OTM.
4. The browser is redirected back to the retrieve incident page, where the modified incident object is displayed with the modifications that were applied.

Incident Delete Example

The incident delete example demonstrates how to delete an incident object.

1. The action ASP scripting calls the function CG_OTMDeleteIncident, passing as parameters the incident object ID, the category ID for this incident, and the timestamp when the incident object was last retrieved. The CG_OTMDeleteIncident function:
   a. Creates an OTMLBOCall object with the method specified as "delete".
   b. Creates an OTMObject for specifying the required fields for the deletion.
   c. Adds the parameters pass in to the OTMObject.
   d. Calls the bExecuteLocal method to request OTM to delete the specified incident object.
2. The browser is redirected back to the retrieve incidents page where the updated list of incidents is displayed.

Accessing Related Business Objects with OTMHelper

The retrieve customer example demonstrates the retrieval of a multiple-business object hierarchy using the OTMHelper classes. Unlike the incident business object used in business object methods examples, the customer business object has associated address and phone list objects. The retrieve customer example shows how to retrieve the customer, primary address, and primary phone information.

The essential elements of this retrieval are:

1. Create an OTMLBOCall object for retrieving the customer information.
2. Add OTMObject parameters for retrieving the customer's phone and address lists.
3. After the OTM call, take the phone and address lists contained in the returned customer OTMRowset object and create individual phone and address OTMRowset objects.

Client-side OTMHelper

The client-side OTMHelper example demonstrates how to retrieve incident information using the cOTMLBOCall ExecuteRemote method.

This example works in a manner similar to the server-side incident retrieve example. However, rather than creating the OTMHelper class objects on the server, the operations are performed by the client browser and the results displayed using DHTML.

Security Example

The security example demonstrates how to secure ASP pages within the OEP architecture in order to prevent unauthorized access. In this example, you attempt to access a secure page. If you have not yet been authenticated, then you are sent to a logon page. After successful authentication, you are allowed access to the secured page.

The example scripting illustrates how to include *otm_common.asp* and call the *initializePageSecurity* function to perform the security check. The example for the security demonstration is in the customization examples directory in the path Examples\Security\Secure.asp.

Result List Control

The result list control sample shows how to create a list using data retrieved from the OEDB. The sample configuration is almost identical to that used in the topic Sample result list. Included are two sample configuration files and the code necessary to embed the list within a Web page. The code shows how to prepare the OEAS source data, configure a list, and handle list events.

The example for the result list control demonstration is in the customization examples directory in the path Examples\ResultList\Sample_result_list.htm.

Programmer's Reference

This book contains detailed information about the OEP COM components, VBScript classes, and application programming interfaces (APIs). Onyx provides numerous public interfaces so that you can customize and extend OEP functionality.

The services provided are divided into the following categories:

Cached data

The Cached data functions provide a means for maintaining application and session data.

Onyx utilities

The Onyx utilities library contains two commonly used functions that are part of the standard Win32 API. These functions have been added to a custom type library to make them easier to use from within OEP.

OTMHelper

The OTMHelper classes provide access to the OEAS Onyx Transaction Manager (OTM). The OTMHelper classes may be used within server-side ASP scripting, or with client-side DHTML.

Result list

There are three client-side JavaScript classes that are used to render some of the lists in OEP.

Security

The Security section describes the functions you call to restrict access to ASP pages, and the OEPSession object.

Cached Data Functions

OEP exposes a number of data caches that reduce the need to make repeated trips to the application database. OEP versions prior to 3.0 employed ASP Application and Session objects for data caching. These cached data APIs provide an abstraction from the underlying implementation of cached data storage.

Cache name	Description
Application	Contains data that is not expected to change while the application is active
Context	Contains data that is likely to change during the application lifetime, or needs to be accessible from the client side
User Preferences	Provides access to a user's preferences information through a single function

Application Cache

The functions that can read and write data from the application cache are listed in the following table. These functions can only be called from server-side ASP scripts.

Function	Description
AppCacheRead	Reads a data item from the application cache
AppCacheWrite	Writes a data item from the application cache
bAppCacheTimedRead	Reads a timed data item from the application cache
AppCacheTimedWrite	Writes a timed data item to the application cache

AppCacheRead Function

The AppCacheRead function retrieves an item from the Application cache.

Syntax

```
Function AppCacheRead(
 sName As String,
  sUserID As String,
 iSiteID As Integer,
 sSessID As String)
As Variant
```

sName

[in] A string that contains the name of the item to retrieve.

sUserID

[in] A string that contains the OEP user ID from the current OEPSession object.

iSiteID

[in] An integer that contains the site ID from the current OEPSession object.

sSessID

[in] A string that contains the session ID from the current OEPSession object.

Remarks

The AppCacheRead function returns the requested item or Null if the item does not exist.

The case of key names is not important and is ignored by the AppCacheRead function.

AppCacheWrite Function

The AppCacheWrite function stores an item identified by a specified name in the Application cache.

Syntax

```
Sub AppCacheWrite(
sName As String,
sUserID As String,
 iSiteID As Integer,
sSessID As String,
 vValue as Variant)
```

sName

[in] A string that contains the name of the item to store.

sUserID

[in] A string that contains the OEP user ID from the current OEPSession object.

iSiteID

[in] An integer that contains the site ID from the current OEPSession object.

sSessID

[in] A string that contains the session ID from the current OEPSession object.

vValue

[in] A Variant that contains the data to be stored.

Remarks

The AppCacheWrite function does not return any status information. If the named item does not exist, it is created. If the named item does exist, it is overwritten.

AppCacheTimedWrite Function

The AppCacheTimedWrite function stores an item identified by a specified name in the Application cache. The item has an expiration time and is removed from the cache when the time expires.

Syntax

```
Sub AppCacheTimedWrite(
 sName As String,
 sUserID As String,
iSiteID As Integer,
```

```
        sSessID As String,
       vValue as Variant,
       lDuration as Long)
```

sName

[in] A string that contains the name of the item to store.

sUserID

[in] A string that contains the OEP user ID from the current OEPSession object.

iSiteID

[in] An integer that contains the site ID from the current OEPSession object.

sSessID

[in] A string that contains the session ID from the current OEPSession object.

vValue

[in] A Variant that contains the data to be stored.

lDuration

[in] A long that identifies when the item expires from the current time. Measured in minutes.

Remarks

The AppCacheTimedWrite function does not return any status information. If the named item does not exist, it is created. If the named item does exist, it is overwritten and assigned the new duration time.

> **Important:** Items placed in the cache with this function must be retrieved using bAppCacheTimedRead. Using AppCacheRead may produce unexpected results.

bAppCacheTimedRead Function

The bAppCacheTimedRead function retrieves an item with an expiration value from the Application cache.

Syntax

```
      Function bAppCacheTimedRead(
       sName As String,
       sUserID As String,
       iSiteID As Integer,
       sSessID As String,
      vValue As Variant)
```

```
    As Boolean
```

sName

[in] A string that contains the name of the item to retrieve.

sUserID

[in] A string that contains the OEP user ID from the current OEPSession object.

iSiteID

[in] An integer that contains the site ID from the current OEPSession object.

sSessID

[in] A string that contains the session ID from the current OEPSession object.

vValue

[in] A Variant that contains the data retrieved.

Remarks

This function returns True if it finds the item and False if the item does not exist or has expired (the item is not actually removed from the cache until this function or bHasAppCacheTimedExpired is called to verify it).

Timed items added to the cache are not automatically updated by OEP. Any item that expires must be readded to the cache using AppCacheTimedWrite.

The case of key names is not important and is ignored by the bAppCacheTimedRead function.

bHasAppCacheTimedExpired Function

The bHasAppCacheTimedExpired function checks on a timed item in the Application cache.

Syntax

```
    Function bHasAppCacheTimedExpired(
    psName As String)
As Boolean
```

psName

[in] A string that contains the name of the item to verify.

Remarks

This function returns True if the item does not exist or has expired (the item is not actually removed from the cache until this function or bAppCachedTimedRead is called) and False if the item is still valid.

Timed items added to the cache are not automatically updated by OEP. Any item that expires must be readded to the cache using AppCacheTimedWrite.

The case of key names is not important and is ignored by the bHasAppCacheTimedExpired function.

Context Cache

The functions that can read and write data from the context cache are listed in the following table. Most of the functions can only be called from server-side ASP scripts, but the last two can be used from within the browser client.

Function	Description
ContextCacheDelete	Removes an item from the context cache
ContextCacheRead	Reads an item in the context cache (server-side)
ContextCacheRead2DArray	Reads an array of items stored in the context cache
ContextCacheReadSubItem	Reads a subitem stored in the context cache (server-side)
ContextCacheWrite	Writes an item to the context cache
ContextCacheWrite2DArray	Writes an array of items to the context cache
ContextCacheWriteSubItem	Writes a subitem to the context cache
jsContextCacheRead	Reads an item from the context cache (client-side)
jsContextCacheReadSubItem	Reads a subitem stored in the context cache (client-side)

A list of keys and subitems for use with the ContextCacheReadSubItem and ContextCacheWriteSubItem functions appears in Context cache keys.

ContextCacheDelete Function

The ContextCacheDelete function removes an item from the Context cache.

Syntax

```
Sub ContextCacheDelete(
 sName As String)
```

sName

[in] A string that identifies the name of the item to delete.

Remarks

The ContextCacheDelete function cannot transform the keys used by the ContextCacheWriteSubItem and ContextCacheReadSubItem functions. To delete an item from the context cache, you must know the abbreviated name of the key. Review the file YourOEPwebsite\common\include\otm_cache.asp for information on how to identify the abbreviated keys.

Primary Navigation

The Navigation Bar and the Header Bar are a part of the basic frame layout of the OEP application. They enable users to select a feature to load either in the Data Area frame or in a separate window.

The **Navigation Bar** works with a two-level tree structure. Top-level elements are known as groups. Elements within groups are known simply as items. The items provide links to customer-related features. The Navigation Bar appears in its own frame on the left side of the main OEP window.

The **Header Bar** appears in the Logo/Graphic frame of the OEP window and contains links to the most commonly used features of the application. Unlike the Navigation Bar, which has a hierarchy and text-only links, the Header Bar has a single row of buttons that identify the link's target.

Configuring the navigation features

Using the Navigation Designer, one of several UCW tools, you can configure Navigation Bar elements, Header Bar buttons, and other UI navigation controls. The Navigation Designer provides a user interface for you to add, modify, and delete the navigation elements. These modifications are automatically applied when you upgrade to future versions of OEP.

Toolbars

OEP uses toolbars extensively to provide functionality at a click of a button, such as to print, save, or delete a record. You can configure toolbars on UCW-enabled OEP pages and you can customize toolbars on other OEP pages.

- With UCW, you can add custom toolbars, add or remove custom toolbar buttons, and change functionality of custom toolbar buttons. For basic instructions, see Configuring toolbars. For a sample UI configuration, see Adding a toolbar or Adding toolbar buttons and other items.
- On OEP pages that cannot be modified using UCW, you can customize toolbars by adding or removing toolbar buttons. For a customization example, see Adding a custom toolbar button.

Database Access

All OEP features access the database through OEAS. For details, see the OEAS Technical Reference.

Install the Onyx Demo Database

Before launching OEP to explore its user interface and understand how it works, consider installing the Onyx Demo Database (Demo.bak) to your Onyx Enterprise Database (OEDB) server.

The Onyx Demo Database gets you up and running quickly by helping you understand how the OEDB is structured and how applications like OEP and OES interact with it. The database is a sample SQL Server 2012 database that contains default data and sample transactional data. Using this database, you can explore each and every feature in OEP.

You can certainly explore OEP without installing the Onyx Demo Database, but if you do you will need to create some records of your own before you can use certain features. For example, you won't be able to view or use product records in OEP until product records exist in the database (to create your own product records, use OES Product Administration).

> **Important:** Use the Onyx Demo Database for instructional purposes only. It is not intended to be used as a development database, because it contains data that you likely won't want to migrate to your production environment.

To learn more about the Onyx Demo Database, read the OEDB Demo Database Readme located on the OEAS product CD.

Explore OEP

Explore OEPBefore modifying OEP, take some time to familiarize yourself with OEP, its many diverse features, and its architecture. The better you understand how OEP works, the easier it will be to modify. And the better you modify OEP to suit your business needs, the more your users will benefit from a rich user experience.

There are many avenues to explore OEP. You can begin your journey by taking any one of them.

Avenues to explore OEP

- **Browse the OEP UI:** Whether you're a business analyst, database administrator, or developer, you may find it most useful to begin by exploring the OEP user interface.

  Log in to OEP using [sa/onyx] logon credentials and navigate through the many features of OEP, including the main customer pages, the various incident categories (sales, service, and support), and the Quick Search tool. Then explore the tools that enable you to interact with these records, including Task Manager, List Manager, and Messenger.

- **Familiarize yourself with OEP's many tools:** Review the list of tools that accompany the OEP and OEAS installations.
- **Peruse the OEP online Help:** Consult OEP's online Help system to learn more about how each OEP feature works.
- **Learn about the system architecture:** As you navigate through OEP and discover more about what OEP is, you may become more interested in how OEP is designed. This guide contains information about the OEP system architecture. For complete details about OEAS and the OEDB, see the OEAS Technical Guide and the OEAS Technical Reference.

  It's important to understand that OEP is just one part of an n-tiered application. N-tier applications are typically divided into three main tiers. The presentation tier, the business logic tier, and the data tier. Keep in mind that although we often discuss OEP as a comprehensive system that spans all three tiers, OEP technically resides only in the presentation tier. The other tiers are comprised of entities that were installed before you installed OEP. The Onyx Enterprise Application Server (OEAS) resides in the business logic tier, and the Onyx Enterprise Database (OEDB) resides in the data tier.

- **Read this guide:** Read all sections of this guide that pertains to you.

The Configuring OEP book is useful for everyone who plans to use UCW to configure OEP.

The Customizing OEP book is useful for developers who plan to write code to customize OEP.

The Administering OEP book is useful for anyone who is configuring, customizing, administering, or maintaining OEP.

- **Peruse the OEAS product documentation:** The OEAS product CD contains several documents and Help systems that provide detailed information about the business logic and data tiers. Because OEP relies so heavily on these tiers, understanding OEAS and the OEDB will greatly enhance your ability to configure, customize, administer, and maintain OEP.
- **Explore OES:** Explore Onyx Enterprise Studio (OES), which is a collection of tools that is installed along with OEAS. If you are responsible for administering and maintaining OEP, you will be most interested in the OES administration tools. If you are a business analyst or developer, you will likely be interested in both the OES administration and design tools. Online Help is available for each OES tool.
- **Learn about other Onyx products:** If you plan to use other Onyx products, such as BRM, EMF, or the SSRS, familiarize yourself with them, too. Install them, navigate their UIs, and read their accompanying product documentation to determine how they may influence the modifications you make to OEP.

  It's typically best to learn about OEP, OEAS, and the OEDB first, and then learn how other Onyx products interact with the OEP/OEAS platform.

Plan Your OEP Implementation

After you understand how OEP works and how each group within your organization can best use it, draft a plan for how you might modify OEP to accommodate each group's unique needs and work flow.

Before you begin to modify OEP, plan your OEP implementation.

Assess your users' needs

Assessing your users' needs—including their daily tasks and the way they accomplish those tasks—is important because it influences how you may want to configure the OEP user interface, and how many different OEP user interfaces you may want to configure. Using Onyx Enterprise Studio (OES), you can then create a separate OEP profile for each OEP user interface that you plan to configure.

Assessing your users' needs is also important because it determines the groups and roles that you create using OES, and how you place each Onyx user within the groups and roles that you define. For details on groups and roles, see Configure users and user access.

Create prototypes for UI design

Before modifying OEP, take the time to thoroughly design and plan the layout, navigation, and behavior for the OEP areas that you plan to modify. Consider creating a prototype of the user interface, and test the prototype by asking your users how they would use it to perform their daily tasks.

Understand which configurations can be accomplished via UCW

It is important to understand which modifications to OEP can be accomplished via UCW, and which require you to write custom code. In some cases, you may need to both configure and customize OEP. All UCW-enabled areas must be configured via UCW. To modify areas not enabled by UCW, however, you will need to write custom code.

Note that you can write custom code within the UCW framework and, if you follow all guidelines for writing this code, it will be safely upgraded with all other UCW modifications when you upgrade to the next version of OEP.

To understand the capabilities of UCW, create a sample OEP profile and design a custom UI, and keep this Help system—and particularly the Configuring OEP book—at your fingertips.

Create a sample OEP profile and design a custom UI

One of the best ways to learn what you can do via UCW is to create a sample OEP profile that you can use for testing purposes, and then design the UI associated with that profile. Don't use the Global profile for testing purposes unless you plan to reinstall OEP to your development environment, because the changes that you make to OEP using the Global profile affect all profile-specific UIs, including the UI associated with the OEP User profile.

For details on creating OEP profiles, see the online Help for OES Security Administration.

Understand the skill set required to make your desired modifications

While many configurations that can be completed using UCW require little more than knowledge about Onyx products and your organization's usage requirements, other configurations may require some programming experience. Carefully consider the skill set of each of your UCW designers and developers to ensure that they are the right person to complete the configuration task.

Type of UCW configuration	Skill set required
Most UCW configurations	• A basic understanding of Web technologies and Boolean logic. • An understanding of UCW actions, conditions, and events. • An understanding of OEAS objects, methods, and properties, which is required for data-binding controls. For details, see the OEAS *Technical Reference*.
Complex UCW configurations that require you to use the Advanced View of Dynamic Forms Designer (see advanced configuration examples)	Some programming experience. In particular, you should be familiar with writing, editing, and debugging JavaScript-formatted scripts and style sheets.

Consider the impact of redesigning page layout

To redesign the layout of an OEP page (or one of its tabs), you can use a UCW tool called Canvas Designer. Canvas Designer enables you to create a canvas (which is comprised of one or more

panels, just as a table is comprised of one or more cells) upon which you place your desired UI elements, such as UI controls, section demarcations, and so on. Once the canvas is created, you can easily move UI controls from one place on the canvas to another. But if you want to make more elaborate changes to page layout, you may need to delete the canvas and completely rebuild it.

Note that the impact of rebuilding a canvas can be quite time consuming, especially if the canvas consists of numerous custom UI controls whose data binding will need to be reset.

Add Domain Data

Before you begin to modify OEP, determine what domain data you want to add to your OEP system. Some of your OEP modifications will likely depend on the domain data that you add to your system.

> **Important:** Each domain data value is assigned a unique ID in the database. It is imperative that these unique IDs are maintained between development, test, and production environments, otherwise unpredictable results may occur. For additional details, see the OEP Installation Guide.

Default installations of OEP include standard domain data such as country codes, region codes, and currency types. It also includes domain data that defines customer types and subtypes (such as Financial/Bank and Financial/Credit Union) and various types of literature that your company may make available to your customers. You will almost certainly want to add the domain data for each product and service that your company offers to your customers, as well as many other types of domain data to support your daily transactions.

You can use OES to create additional domain data, or to modify existing domain data to make it more relevant to your business. For details about the various OES tools, see the OEAS Technical Reference.

Configure Your Development Environment

Before you can use UCW to configure OEP, you must configure your development environment by:

1. Creating custom OEP profiles, if desired
2. Creating user accounts for each person who intends to use UCW
3. Adding sample data to your database

Create Custom OEP Profiles

Only one OEP profile is provided with the installation of OEP: the OEP profile named "OEP User".

If you plan for all of your OEP users to use the same OEP user interface (UI), you can apply all your configurations to the OEP User UI accessed with this profile. If you plan to configure multiple versions of the OEP UI, however, you must create additional OEP profiles.

Note: Users who have access to multiple UIs select which OEP profile they want to use when logging on to OEP. To access another custom UI, users must close OEP and then log on again selecting the related OEP profile.

Each OEP profile that you create enables you to configure a custom UI. The UI's appearance, behavior, and other such characteristics depend on the modifications made to both it and the baseline UI.

For detailed steps on creating OEP profiles, see the online Help for OES Security Administration.

Create User Accounts for UCW Designers

To use UCW, UCW designers must have an Onyx user account with permissions to the UI.OEP.UCW.Configure resource.

A single UCW designer can log on to OEP using the standard Onyx system administrator account [SA/onyx] to use UCW. If several UCW designers plan to work at the same time, however, they must have separate Onyx user accounts, each with permissions to the UI.OEP.UCW.Configure resource.

Note: For detailed instructions on completing the procedure below, see the online Help for OES Security Administration.

To create user accounts for UCW designers:

1. Launch OES Security Administration.
2. If necessary, create an Onyx user account for each UCW designer.
3. Give the UCW designer access to the required profile(s).

To give the UCW designer access to:	Assign the UCW designer's user account to:
The Global profile (but not the OEP User profile)	The UI:OEP.ucw.configure resource
Both the Global profile and the OEP User profile	The OEP.administrator role **Note:** This role also grants access to several other OEP resources

4. Give the UCW designer access to the required OES tool(s).

To give the UCW designer access to:	Assign the UCW designer's user account to:
Specific OES tools (for example, the Security Administration tool for creating OEP profiles, or the Reference Table Administration tool for	The UI resources that provide access to the

To give the UCW designer access to:	Assign the UCW designer's user account to:
configuring UDFs)	required tools
All OES tools	The Administrator role

5. Associate each UCW designer's user account to custom OEP profiles, if desired. Doing this allows UCW designers to log on to OEP using that OEP profile, enabling them to configure the UI associated with that OEP profile.
6. If you're using integrated Windows authentication within your development environment:
   - On your OEP Web server, use Computer Management (a Windows administration tool) to add the Windows account of each UCW designer to the OnyxUCWUsers group.

Add Sample Data to Your Database

Accessing UCW is as easy as navigating to the UCW-enabled area of OEP that you want to configure and clicking  . To test certain pages that you configure, however, data must exist in the database. For example, a company record must exist before you can test the Company Edit page. Therefore, it's a good idea to add a few records to your database before you begin to configure OEP.

Using OES Product Administration, add the following data to the OEDB:

- a product
- an incident product for a sales incident
- an incident product for a service incident
- an incident product for a support incident

Using OES Reference Table Administration, add lookup values for the following items:

- type (The type of the incident, such as Recurring or Single Occurrence. This is different than an incident category.)
- status (The status of the incident, such as Open or Closed.)
- priority (The priority of the incident, such as High, Medium, Low.)
- source (The source of the incident, such as Direct or Referral.)

Using OEP, add at least one record of each of the following types:

- company
- individual
- product
- a sales incident associated with either a company or individual record
- a service incident associated with either a company or individual record

- a support incident associated with either a company or individual record
- a task associated with any incident record that you created

Modify OEP

After you've planned your OEP implementation and configured your development environment for use, you're ready to modify OEP.

Although your organization may be perfectly happy using the default implementation OEP, you may want to make certain modifications to the way OEP looks and behaves. OEP is fully modifiable and extensible.

Note: To modify how OEP behaves, you may need to modify each tier that comprises the OEP system architecture. For details on modifying OEAS and the OEDB, see the OEAS Technical Reference and the OEAS Technical Reference.

To modify OEP:

1. Determine whether your modifications would be considered configurations or customizations.
2. Determine how you want to implement your changes using UCW, OES, and other Onyx tools.

   - Using UCW, you can configure the UI design and behavior of many OEP feature areas without writing complex code.
   - Using OES, you can make various other modifications that help define how OEP behaves.

   For instance, you can enable and disable the Forecast & Quotes feature of OEP simply by changing a system parameter. Or you can do something more complex, such as create campaigns for your Marketing department or create UI scripts for use by your call center. The *OEAS Technical Reference* contains detailed information about each OES tool.
3. Review our best practices and customization guidelines to avoid potential problems.
4. Use UCW to configure UCW-enabled OEP areas.

   - To make common UI modifications to all OEP profiles, log on to OEP using the Global profile.
   - To make profile-specific UI modifications, log on to OEP using the OEP User profile or any other OEP profile that you created.
5. Customize other OEP areas as desired.
6. Compile your UCW configurations.
7. Unit test your modifications.
8. Migrate your changes to your test environment.
9. Thoroughly test all changes you made to OEP.
10. Migrate your changes to your production environment.

11. Test OEP in your production environment.
12. Deploy OEP to your users.

Test OEP

To ensure that OEP functions as expected, be sure to test every modification you make to OEP. Consider performing at least one test pass for each custom UI you configured.

We recommend that you perform full system testing using a separate test environment rather than in your development environment. If you haven't already set up your test environment, do so now by following the instructions provided in the OEP Installation Guide.

Before testing OEP:

- Configure users and user access
- Migrate your modifications to your test environment

Install OEP to Your Production Environment

When you're finished modifying OEP and testing your changes, you're ready to install OEP to your production environment.

Note: If you plan to maintain an entirely separate development environment for future modifications to OEP, you may also need to install OEAS and the OEDB to your production environment, as well.

To install OEP to your production environment, follow the setup instructions provided in the *OEP Installation Guide*.

Configure Users and User Access

After migrating all OEP modifications to your production environment and verifying that everything functions as expected, you're ready to add OEP users and configure their access permissions to OEP features.

OES Security Administration enables you to create Onyx user accounts and define each user's access permissions. You may want to give certain users permission to access all OEP records and to be able to use all of OEP's functionality, and you may want to restrict other users' access permissions so that they'll be able to use only a limited set of OEP's records and functionality. For instance, you may not want users outside of your sales and finance groups to have access to records that reflect what customers pay for your products and services.

Fortunately, you don't need to set each user's access permissions separately. Instead, you can use Security Administration to create roles that define a specific set of access permissions, and then assign users to these roles as appropriate.

Adding users

Use OES Security Administration to add users to the Onyx database. For detailed instructions, see the OES online Help.

Note: Before adding users to your database, you may want to contact Onyx Product Support to determine if you need to purchase additional Onyx licenses.

To add users:

1. Launch OES Security Administration.
2. Create one or more groups, as desired.

   You typically assign Onyx users to groups. Groups are hierarchical classifications of users, and may or may not correspond to actual departments or teams within your organization. Note that users can belong to only one group.

3. Create one or more roles, as desired, and define each role by granting them access to various resources.

   Onyx users can be assigned to one or more roles. Roles, like groups, may or may not correspond directly to actual roles within your organization. Here, roles pertain to OEP and to the role(s) of each user who uses OEP. You can use roles as a means to manage security and access to data, as well as a means to drive business rules. Users can belong to any number of roles.

   For example, many users in your organization will likely share the same set of access permissions, since they will generally use the same set of OEP functionality. Other users in your organization, however, such as those in your Finance and Sales departments, may require additional access permissions to OEP functionality. And yet another set of users, such as OEP administrators, database administrators, and UCW designers, require access permissions not only to the standard set of OEP functionality, but also to OES, OPS, and possibly other tools, as well. You can also use roles to drive business rules, such as by sending a sales contract to your legal advisor when you approach the final stage of a sales deal.

   OEP provides several default roles to which you can assign users. You may, however, want to create entirely custom roles to suit your needs. Configure each role by granting or denying access to specific resources.

   Note that each role that you create must be granted permission to the “UI:OEP.Logon” resource, either directly or by membership in a role with permissions to that resource. Users without permission to that resource cannot log on to OEP. You may also want to assign each role to at least one OEP profile (unless you want to assign individual users to their appropriate OEP profile), since users must be assigned to at least one OEP profile in order to log on to OEP.

4. Create the appropriate Onyx user accounts for each group that you created earlier.

When creating an Onyx user account, be aware of certain OEP account limitations.

For each user, you must indicate whether the user account will be authenticated using Onyx authentication or integrated Windows authentication.

5. Add each user to one or more roles, as desired.
6. Grant/deny each user to additional resources, to either permit or restrict their access to additional functionality.
7. Ensure that each user is assigned to at least one OEP profile, either directly or by membership in a role that is assigned to at least one OEP profile, otherwise they cannot log on to OEP.

Migrate Your Modifications

After installing OEP to your production environment, migrate all desired modifications from your development environment to your production environment.

To migrate your modifications, begin by migrating all data tier (OEDB) modifications first. Then migrate all modifications that you made to your business logic tier (OEAS), and finally migrate all modifications that you made to your presentation tier (OEP). For a visual depiction of the migration effort, see the graphic at the bottom of this page

Consider using a source code control application (such as Visual Source Safe or TFS) to help manage these tasks.

Note: If you're upgrading from a previous version of OEP, or if you're deploying your modifications in iterations, be sure to deploy only the necessary/desired modifications and take care not to unintentionally overwrite existing data or components that you want to keep.

Migrate OEDB modifications

Migrate all modifications to the OEDB first.

1. From your development environment to your production environment, migrate all desired database objects, including all custom stored procedures, retrieveList stored procedures, views, tables, and SQL functions. Then compile these objects, as necessary, in your production database.

   For tables, be sure to migrate the definitions of foreign keys and indices as part of the table definition. Also make sure that your scripts differentiate between modifying an existing table (using an ALTER TABLE script) and adding a new table (using a CREATE TABLE script).

2. Migrate all domain data, including all reference parameters and national language (NatLang) messages, from its master database to your production environment. Make certain that the unique ID of each domain data value remains unchanged. You may want to write scripts to accomplish this.

3. If you chose to set default user preferences in your development environment, either migrate all default user preferences to your production environment (you may want to write a script to accomplish this), or re-implement them in your production environment.

Migrate OEAS modifications

After you have migrated all modifications to the OEDB, proceed to migrate your changes to OEAS.

1. From your development environment to your production environment, migrate all custom OEAS step components by copying each compiled DLL to your production OEAS server and registering them with COM+.
2. Copy the OED (working copy) from the OEAS server in your development environment to the appropriate location on the OEAS server of your production environment. Then use OES Object Designer to publish the OED.

   If your production environment uses more than one OEAS server, use the publish.vbs script that's included with the OES Object Designer to copy the working copy of the OED to your production environment. You must complete all four steps in the script on one (and only one) of the OEAS servers, and then perform all steps except the stored procedure compilation step on your other OEAS servers.

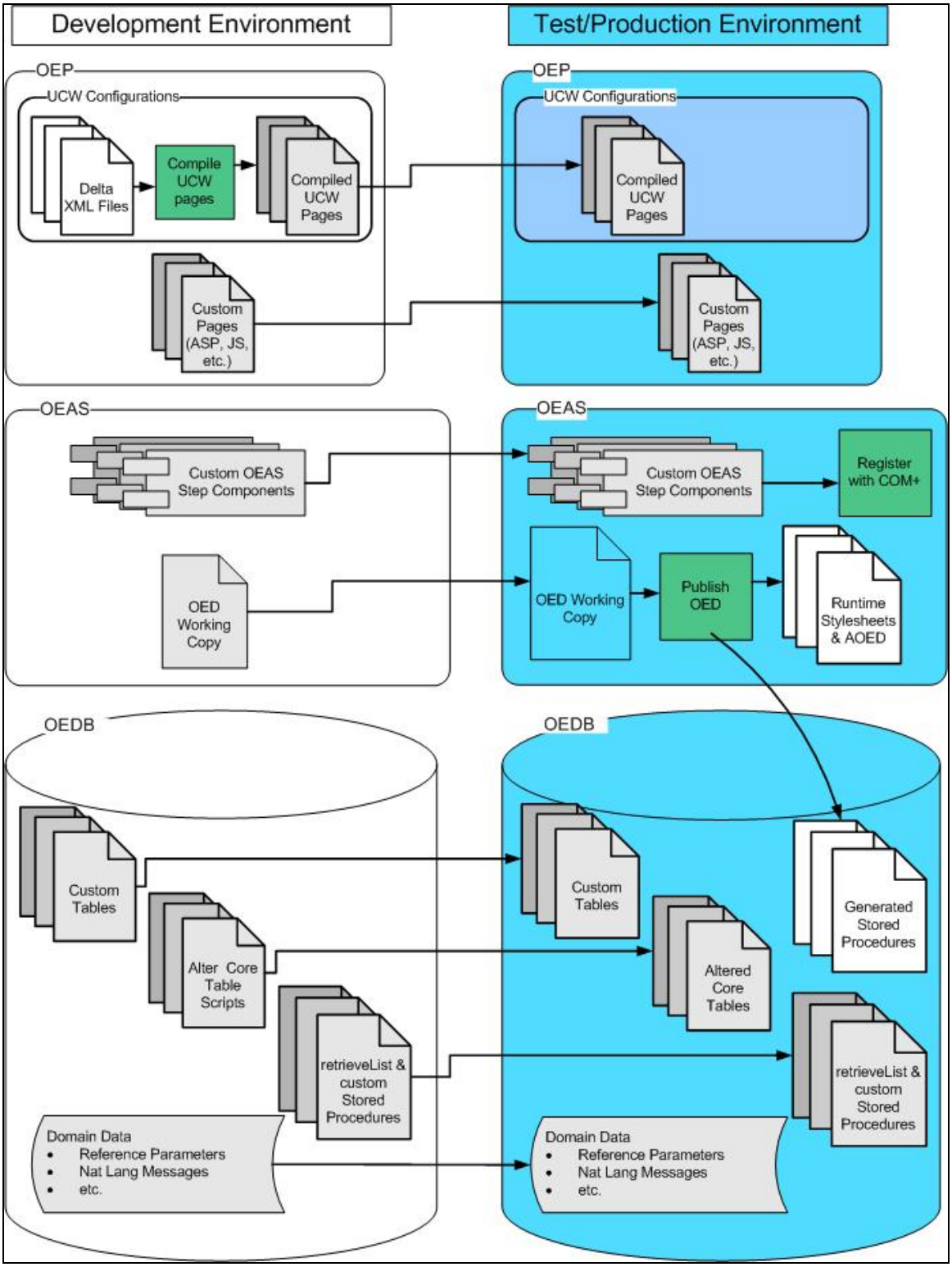
Migrate OEP modifications

After you have migrated all modifications to OEAS and the OEDB, migrate all modifications that you made to OEP (and to any other user interface that you have designed to use the Onyx platform).

1. From your development environment, migrate to your production OEP server all custom OEP pages (ASP pages, JavaScript pages, and so on) that you customized without using UCW by copying them to the appropriate OEP directories.
2. From your development environment, migrate to your production OEP server all OEP pages that you configured using UCW (be sure that your modifications are compiled first).
3. After migrating all modifications to your production environment:

   - Stop and restart your OEP Web server, such as by using the Internet Information Services (IIS) Manager tool or by executing the IISReset command (via Command Prompt).
   - Test your OEP system to ensure that everything functions as expected.

Graphical depiction of migration effort



Set Default User Preferences

Using a special user account called *User Preferences Default*, you can establish settings that help to define how OEP behaves for your various users. Users can later override these settings to personalize OEP according to their own preferences.

Although you can set default user preferences in the development environment and later migrate these preferences to your test and/or production environments, it's typically easier to implement them only in your test and/or production environments.

You can set default user preferences for all custom UIs simultaneously, or you can implement different default user preferences for each separate OEP profile that you've created.

Deploy OEP

After you've prepared your OEP production environment for use, you're ready to deploy OEP to your users.

Deployment suggestions

- Before deploying OEP to your users, consider offering an training course to get your users up and running quickly
- Inform your users of our recommendations for client OEP users and what to expect when printing and emailing UCW-configured pages
- If you're upgrading from an earlier version of OEP, inform your users that they must clear their Internet Explorer caches of OEP data before launching the new version of OEP
- Provide each OEP user with the following logon credentials:
  - Logon ID and password (not necessary if you're using integrated Windows authentication)
  - OEP profile(s)

Administer OEP

Congratulations—you've successfully implemented and deployed OEP!

You are now in the mode of administering OEP for your users and monitoring any enhancement requests that your users may make for future modifications to OEP. You may also be responsible for maintaining the Onyx database, as well.

For details about administering OEP, see Administering OEP. For details about maintaining OEAS and the OEDB, see the *OEAS Technical Reference*.

Changing Password for Services

When you run the OEAS Config, the passwords you enter for OGS and ONS connectivity to the database, as well as to the EWS and SMTP server are stored in encrypted form in the respective configuration files. Use the Password Encrypt utility included with your installation to change these passwords after installation.

To change the password:

1. In your Onyx installation package, navigate to Utilities and copy the PassEncrypt folder to your OEAS server.
2. Ensure that the privatekey.pvk and publickey.puk files are Read Only.
3. On your OEAS server, open Command Prompt and change the directory to the copied location.
4. Run the following command to generate the encrypted password.
   - Replace [PUK path] with the path where you copied the publickey.puk file.
   - Replace [password] with the new password in plain text.

   ```
   Passencrypt.exe /e /puk "[PUK path]" /p [password in plain text] /pr yes
   ```
5. Copy the encrypted password which is printed below the caption Encrypted Password.
6. Navigate to your OEAS installation folder, and open the configuration file for which you want to change the password.
7. Under the Password attribute, paste the encrypted password that you copied from Command Prompt.
8. Save and close the file.

Modifying OGS Configuration File

You can define how information is displayed in Navigator and in Onyx Mobile by changing specific settings in the OGS Configuration file. Use this file to specify:

- The system limit for the number of records that are displayed when you run a search in Navigator
- The time limit in milliseconds, after which any request from Onyx Mobile to OGS will time out.

To modify OGS settings:

1. On the OEAS server, navigate to the OnyxGatewayService install location, and open the `OnyxWindowsService.exe.config file`. The default location is `..\Program Files\Onyx\AppServer\Applications\Onyx\OnyxGatewayService\OnyxWindowsService.exe.config`
2. To change the limit for number of records to display in Navigator, search for the attribute `<add key="MaxRecordsLimit" value="999" />`, and change the value as desired. The maximum system limit corresponds to the maximum possible value for the data type integer, which is 2,147,483,647.
3. To change the time limit for requests from Onyx Mobile, search for the attribute `<add key="RequestTimeOut" value="180000" />`, and change the value as desired. This value represents the time limit in milliseconds.
4. Save the `OnyxWindowsService.exe.config` file.

Twitter Integration

Onyx 7.8 supports v1.1 APIs of Twitter. In the **Social Media** tab of the Onyx application, a user can enter a twitter handler in the Identifier field.

To register for the Twitter app

1. Go to dev.twitter.com and login with your Twitter account credentials.
2. In the Twitter home page, select **My applications** from the user's drop-down list.
3. Click on **Create a new application**.
4. Type the relevant information in the Twitter application, accept terms/conditions and click on **Create your Twitter application**.
5. From the app screen, click on the **Details** tab. The **OAuth Settings** section of the tab will display these two fields:
   - Consumer Key
   - Consumer secret
6. Copy the above values from the **OAuth Settings** to the OGS Config file for the following tags:
   - Consumer Key - token_ConsumerKey
   - Consumer secret - token_ConsumerSecret
   - twitter_Count – No. of tweets per request (max value 200)

Understanding the configuration changes for Twitter

Onyx has multi-site support for Twitter. To enable the multi-site support, add _siteid (Id of the site) to the following config settings:

- Consumer Key - token_ConsumerKey_siteId
- Consumer secret - token_ConsumerSecret_siteId
- twitter_Count _siteId– No. of tweets per request (max value 200)

In case there is no siteid settings present, it will take the value from default settings. For each siteid, it is recommended to have different app tokens to increase the upper limit for Twitter APIs. For Onyx we use the following REST APIs v1.1.

- GET stateuses-user_timeline
- GET users-show

For more information see https://dev.twitter.com/rest/public.

OEP UI Security Resources

Using OES Security Administration, you can secure access to OEP and its various features. By granting or denying access permissions to UI resources, you define the actions that your OEP users can perform. Permissions that you set for a user action are applicable both, when the user performs the action through the respective OEP page as well as through Navigator.

Set access permissions to UI resources in order to:

- Grant or deny access to OEP
- Grant or deny access to an OEP feature, such as to scripted workflows
- Grant or deny permission to perform certain actions for an OEP feature, such as by denying users the permission to delete company and individual records

It's good practice to secure UI resources at the role level, and then set more restrictive permissions for users within each role.

Note: There are no resources for Forecasts and Quotes.

Resource used for OEP logon

The following table identifies the security resource for connecting to OEP.

Resource Name	Function
UI:OEP.Logon	This resource provides access to OEP. Users without permission to this resource cannot log on to OEP.

Resources used for alternate addresses

The following table identifies the security resources for managing customers' alternate addresses.

This Resource...	Allows users to...
UI:OEP.address.delete	Delete alternate addresses.
UI:OEP.address.update	Modify existing alternate addresses.

Resources used for attachments

The following table identifies the security resources for attachments (files attached to customer records).

This Resource...	Allows users to...
UI:OEP.attachment.delete	Delete attachments.
UI:OEP.attachment.insert	Upload (add) attachments.
UI:OEP.attachment.retrieve	Download and view attachments.

This Resource...	Allows users to...
UI:OEP.attachment.update	Modify existing attachments.

Resources used for campaigns

The following table identifies the security resources for campaigns.

This Resource...	Allows users to...
UI:OEP.campaign.delete	Delete campaigns.
UI:OEP.campaign.insert	Create new campaigns.
UI:OEP.campaign.retrieve	View campaigns.
UI:OEP.campaign.update	Modify existing campaigns.

Resources used for companies

The following table identifies the security resources for company records.

This Resource...	Allows users to...
UI:OEP.company.delete	Delete company records.
UI:OEP.company.insert	Add new company records.
UI:OEP.company.retrieve	View company records.
UI:OEP.company.update	Modify existing company records. It also allows users to clone company records.
UI.OEP.merge.company	Merge two existing company records using Navigator.

Resources used for contacts

The following table identifies the security resources for both external contacts and internal contacts.

This Resource...	Allows users to...
UI:OEP.externalContact.delete	Delete external contacts.
UI:OEP.externalContact.insert	Add new external contacts.
UI:OEP.externalContact.retrieve	View external contacts.
UI:OEP.internalContact.delete	Delete internal contacts.
UI:OEP.internalContact.insert	Add new internal contacts.
UI:OEP.internalContact.retrieve	View internal contacts.

Resources used for emails

The following table identifies the security resources for email records.

This Resource...	Allows users to...
UI:OEP.emailMessage.delete	Delete email records.
UI:OEP.emailMessage.insert	Add new email records.
UI:OEP.emailMessage.privateAccess	Set email records to private access, meaning that only the user who set access to 'private' can view that email record.
UI:OEP.emailMessage.readOnly	Set email records to read-only access, meaning that the email record cannot be deleted by other users.
UI:OEP.emailMessage.retrieve	View email records.
UI:OEP.emailMessage.update	Modify existing email records.

Resources used for email templates

The following table identifies the security resources for email templates.

This Resource...	Allows users to...
UI:OEP.mergeTemplate.delete	Delete email templates.
UI:OEP.mergeTemplate.insert	Add new email templates.
UI:OEP.mergeTemplate.retrieve	View email templates.
UI:OEP.mergeTemplate.update	Modify existing email templates.
UI:OEP.mergeTemplate.update.public	Share email templates within OEP, making them available to other OEP users.

Resources used for incidents

The following table identifies the security resources for sales, service, and support incidents.

This Resource...	Allows users to...
UI:OEP.incident.delete.sales	Delete sales incidents.
UI:OEP.incident.delete.service	Delete service incidents.
UI:OEP.incident.delete.support	Delete support incidents.
UI:OEP.incident.insert.sales	Add new sales incidents.
UI:OEP.incident.insert.service	Add new service incidents.
UI:OEP.incident.insert.support	Add new support incidents.
UI:OEP.incident.retrieve.sales	View sales incidents.
UI:OEP.incident.retrieve.service	View service incidents.
UI:OEP.incident.retrieve.support	View support incidents.
UI:OEP.incident.update.sales	Modify existing sales incidents.

This Resource...	Allows users to...
UI:OEP.incident.update.service	Modify existing service incidents.
UI:OEP.incident.update.support	Modify existing support incidents.
UI:OEP.incident.batchUpdate.sales	Modify a batch of existing sales incidents using Navigator.
UI:OEP.incident.batchUpdate.support	Modify a batch of existing support incidents using Navigator.
UI:OEP.incident.batchUpdate.service	Modify a batch of existing service incidents using Navigator.

Resources used for individuals

The following table identifies the security resources for individual records.

This Resource...	Allows users to...
UI:OEP.individual.delete	Delete individual records.
UI:OEP.individual.insert	Add new individual records.
UI:OEP.individual.retrieve	View individual records.
UI:OEP.individual.update	Modify existing individual records. It also allows users to clone individual records.
UI.OEP.merge.individual	Merge two existing individual records using Navigator.

Resource used for List Manager

The following table identifies the security resource for List Manager.

This Resource...	Allows users to...
UI:OEP.query.update.public	Create new List Manager queries and make them available to other OEP users. It also allows users to modify other public List Manager queries.

Resources used for literature

The following table identifies the security resources for literature.

This Resource...	Allows users to...
UI:OEP.literature.delete	Delete literature items.
UI:OEP.literature.insert	Add new literature items.
UI:OEP.literature.retrieve	View literature items.
UI:OEP.literature.update	Modify existing literature items.

Resources used for Messenger

The following table identifies the security resources for Messenger.

This Resource...	Allows users to...
UI:OEP.messenger.delete	Delete Messenger messages.
UI:OEP.messenger.insert	Create and send Messenger messages.
UI:OEP.messenger.retrieve	View Messenger messages.

Resources used for common Navigator functions

The following table identifies the security resources for controlling various common functions in Navigator.

This Resource...	Allows users to...
UI:OEP.searchFilters.configure	Configure Navigator search criteria.
UI:OEP.Navigator.PublicFolder.ShowHide	Access public queries folder, save queries, and run saved public queries.
UI:OEP.Navigator.PrivateFolder.ShowHide	Access private queries folder, save queries, and run saved private queries.
UI:OEP.resultgrid.configure	Configure Navigator result grid columns.
UI:OEP.resultGrid.print	Print Navigator search results.
UI:OEP.resultGrid.export.csv	Export Navigator search results to a CSV file.
UI:OEP.resultgrid.configure.maxRecord	Modify the value for maximum search records to display in Navigator.
UI:OEP.Navigator.PublicQuery.Manage	Delete public queries, clone public queries, rename public queries, edit public queries, add public folders, rename public folders, delete public folders, and move public and private queries.
UI:OEP.reminder.insert	Add reminders using Navigator Grid Action Menu.
UI:OEP.alert.insert	Add alerts using Navigator Grid Action Menu.
UI:OEP.script.retrieve	View Script as an entity in the Search Type drop-down of the Navigator Search Criteria pane.
UI:OEP.batchupdate.task	Perform a batch update for Task records.
UI:OEP.batchupdate.workticket	Perform a batch update for Work Ticket records.
UI:OEP.appointment.gam UI:OEP.company.gam UI:OEP.email.gam UI:OEP.forecast.gam UI:OEP.individual.gam UI:OEP.product.gam	Perform GAM actions for the corresponding entry.

This Resource...	Allows users to...
UI:OEP.incident.sales.gam UI:OEP.script.gam UI:OEP.incident.service.gam UI:OEP.incident.support.gam UI:OEP.task.gam UI:OEP.workticket.gam	
UI:OEP.task.retrieve	View Task as an entity in the Search Type drop-down of the Navigator Search Criteria pane.
UI:OEP.appointment.retrieve	View Appointment as an entity in the Search Type drop-down of the Navigator Search Criteria pane.

Resources used for products

The following table identifies the security resources for products.

This Resource...	Function
UI:OEP.customerProduct.delete	Delete products.
UI:OEP.customerProduct.insert	Add new products.
UI:OEP.customerProduct.retrieve	View products.
UI:OEP.customerProduct.update	Modify existing products.
UI:OEP.customerProductLine.delete	Delete product line items.
UI:OEP.customerProductLine.insert	Add new product line items.
UI:OEP.customerProductLine.retrieve	View product line items.
UI:OEP.customerProductLine.update	Modify existing product line items.

Resource used for the results list control

The following table identifies the security resource for the results list control (the table, or list, that displays the results of a search query). This setting affects all instances of the results list control throughout OEP.

This Resource...	Function
UI:OEP.RLControl.update	Configure results lists that appear in OEP, such as by reordering columns and changing column widths.

Resources used for scripted workflows

The following table identifies the security resources for scripted workflows, which display as a series of Wizard-like pages known as a script session.

This Resource...	Function
UI:OEP.scriptSession.delete	Delete a script session.
UI:OEP.scriptSession.insert	Launch a script session and save data to the database.
UI:OEP.scriptSession.retrieve	View completed (saved) script sessions.
UI:OEP.scriptSession.update	Modify an existing script session.
UI:OEP.scripting.designer	Use OES to design and test unpublished script sessions.

Resources used for surveys

The following table identifies the security resources for surveys.

This Resource...	Function
UI:OEP.survey.delete	Delete surveys.
UI:OEP.survey.insert	Add new surveys.
UI:OEP.survey.retrieve	View surveys.
UI:OEP.survey.update	Modify existing surveys.

Resources used for tasks

The following table identifies the security resources for sales, service, and support tasks.

This Resource...	Function
UI:OEP.task.delete.sales	Delete sales tasks.
UI:OEP.task.delete.service	Delete service tasks.
UI:OEP.task.delete.support	Delete support tasks.
UI:OEP.task.insert.sales	Add new sales tasks.
UI:OEP.task.insert.service	Add new service tasks.
UI:OEP.task.insert.support	Add new support tasks.
UI:OEP.task.retrieve.sales	View sales tasks.
UI:OEP.task.retrieve.service	View service tasks.
UI:OEP.task.retrieve.support	View support tasks.
UI:OEP.task.update.sales	Modify existing sales tasks.
UI:OEP.task.update.service	Modify existing service tasks.
UI:OEP.task.update.support	Modify existing support tasks.

Resource used for UCW

The following table identifies the security resource for UCW.

This Resource...	Function
UI:OEP.ucw.configure	Use UI Configuration Workbench (UCW) to configure the OEP UI for one or more OEP profiles. Members of the OEP.administrator role have permissions to this resource. To use UCW, OEP must be installed to a development environment.

Resources used for work tickets

The following table identifies the security resources for work tickets.

This Resource...	Function
UI:OEP.workticket.delete	Delete work tickets.
UI:OEP.workticket.insert	Add new work tickets.
UI:OEP.workticket.retrieve	View work tickets.
UI:OEP.workticket.update	Modify existing work tickets. It does not, however, allow (or restrict) clone functionality.

Logging on to Onyx Mobile

When you launch Onyx Mobile for the first time on your mobile device, you must enter the configuration settings provided by your administrator before you can log on.

To log on to Onyx Mobile:

1. On your iOS or Android device, launch Onyx Mobile.

   When you launch the app for the first time, the Onyx Configuration screen appears.

2. Enter the configuration settings provided by the administrator.

   - URI Name: Type the URI address of the OGS/OGS Proxy.
   - Application Name: Type the OEAS application name.
   - Site ID: Type the Onyx site ID.
   - User ID: Type your Onyx user name.

3. Tap Save. The configuration is saved and the Login screen appears.
4. Accept the default user name or type a different one, and then type your Onyx password.
5. Tap Login to access Onyx Mobile.

   To change previously entered configuration settings, tap Settings on the Login screen, enter the new settings on the Onyx Configuration screen, and tap Save.

4

Notifications and Calendar Appointments

Overview

Use this information to configure and customize Calendar Appointments and Notifications to suit your organization's requirements.

- Notifications
- Calendar Appointments

Notifications

The Notifications feature enables users to receive email notifications when specific changes are made to Onyx records. Users can subscribe to notifications that are relevant to their work and can also forward them to other users.

Onyx integrates with Event Management Framework (EMF) to provide ten default notifications that address common business needs. EMF is an Aptean solution that monitors events in Onyx and sends email notifications to subscribed users. Using EMF, you can define schedules to process the delivery of these notifications and modify the notification email templates to control the information that users receive in their email. You can also add custom notifications based on your organization's needs.

Before you can begin using the Notifications feature, you must install and configure EMF to be able to connect with Onyx. For information on installing and configuring EMF, see the EMF documentation.

The tasks explained in this topic are:

- Creating Notifications
- Configuring Notification Processes
- Configuring Notifications
- Enabling Notifications
- Disabling Notifications
- Enabling Specific Notifications
- Disabling Specific Notifications
- Creating LBO Adapters Using Step Component
- Adding UI Resource Permission
- ONS Extensions
- Defining Names for Custom Notifications
- Enabling/Disabling SQL Triggers
- Default Notifications

Creating Notifications

Currently notifications work on a multiple logical object. For example, to create a new notification type on Logical Object “obj”

1. Make sure LBOAdapter is enabled in LBOAdapter.config file in the path ..\Program Files\Onyx\AppServer\Components\LBOAdapter.config on the object “obj”.
2. Add notification type in persistence DB.
3. If the secondary id of new notification type is for example “25”. Create an entry into “on_ summary_email_subject” table with notification_id = 25.
4. Create a new process with the name “MyNewNotification”.
5. Edit the link between “Event Id” and “Return Accepted” modules in “OnyxNotificationBaseProcess” and append "OR" condition (you can copy it from other conditions). Edit “Against this expression(value)” field with “/25”.
6. Open “OnyxNotificationProcess_Execute” process and call “MyNewNotification” process from “Case Statements” module. Copy link properties of other notification types and paste to this link. And update “Against this expression(value)” field with "/25".
7. Enter "call parameter" values inside call parameters tab(copy the values from other existing process).

Adding custom notifications

Based on your organization's requirements, you can create custom notifications to alert users about specific changes to Onyx records. Additionally, if you want to add any custom fields to the parameters or to the email template of an existing notification.

Your installation package includes a folder LBOAdapterTemplate to help you create a custom notification. This folder is located under Customization Support\Application Server (OEAS)\LBOAdapterTemplate.

> **Important:** To create a custom notification, refer to the folder in the following location Customization Support\Application Server (OEAS)\LBOAdapterTemplate.

To create a custom notification, refer to the folder in the following location Customization Support\Application Server (OEAS)\LBOAdapterTemplate.

To create a custom notification, you must:

Add Event Notifications

Adding Event

You can add an Event to a notification by adding data to the Event and EventMI tables.

1. Go to the Event table in persistence database.
2. Add the appropriate LboObjectId that is present in LboObject table.

3. If there is no related LboObjectId found for the event, then first enter data for the Lbo Object in LboObject table.

The following is an example to add a Support Incident Event

In the Event table:

1. Generate an EventId(GUID).
2. Enter the SiteId and SecondaryId a unique value.
3. Enter the LboObjectId(Foreign key from LboObject table)
4. Enter the RecordStatus (If the event needs to be active make it "true")
5. Fill in the common columns of UpdateBy, UpdateDate, InsertBy, and InsertDate.

In the EventMl table:

1. Add the EventId (foreign key from Event table).
2. Enter the SiteId, add LanguageCode specific to the application and the EventName(description of the event).

Adding Change Type

You can add a change type to a Notification by making changes in the ChangeType table in persistence database. The change type indicates the type of action to be performed for each event.

For example, to add a change type (Insert) to a Support Incident, go to the ChangeType table and add Insert to the change type record. In an OOB setup, the change types available are Insert, Update and Delete.

For new change type data need to be entered in following tables:

ChangeType

Associate Change Type to Event

You can combine Event Type and Change Type by adding data in the EventChangeType table. The data tells us what are the change types associated with each event.

Adding filters to Event

You can add specific Filters to EventFilter table in persistence database. EventId is the foreign key of Event table.

Updating Notification Type Table

To add a new notification in OEP, you must add a new record in the NotificationType and NotificationTypeMl tables in Persistence database. In these tables NotificationType is the master table, so data need to be entered first in the NotificationType table before being entered in the NotificationTypeMl table.

Use the following information to update the NotificationType table for your custom notification.

Column Name	Description
NotificationTypeId	Enter a unique GUID for the notification reference.
SiteId	Enter the Onyx site ID.
SecondaryId	Enter the Secondary ID for the notification. This is a unique value for each row in the table.
NotificationDeliverTypeId	Enter the Primary Key of the NotificationDeliveryType table to specify whether to deliver an individual or a summary email for the notification.
FromEmail	Enter the value that will appear in the From email address field when sending emails.
UIResourceName	Enter the security resource permission.
RecordStatus	Enter 1 for the notification to be available for selection in the Notifications Subscription window. Enter 0 to hide the notification in the Notification Subscription window. the value 0 is also used for calendar appointments in this table, because they do not appear in the Notification Subscription window.
BrokerUrl	The data in this column is used for communication between ONS and EMF for a specific Notification.

Updating the NotificationTypeMl table

Use the following information to update the NotificationTypeMl table for your custom notification.

Column Name	Description
NotificationTypeId	Enter a unique GUID for the notification reference.
SiteId	Enter the Onyx site ID.
LanguageCode	Enter the code for the installation language from the language table. ENG indicates English and JPN indicates Japanese.
NotificationTypeName	Enter the name for the custom notification that will appear in the drop-down list in the Notification Subscription window.
NotificationDescription	Enter a brief description for the notification that explains the notification's purpose.

Sample script to update the NotificationType and NotificationTypeMl tables

Use the following sample script to update the tables for new notifications.

```
declare @NotificationDeliveryTypeId uniqueidentifier
declare @NotificationTypeId uniqueidentifier
select @NotificationDeliveryTypeId = NotificationDeliveryTypeId
from NotificationDeliveryType
where secondaryId=2
set @NotificationTypeId=NEWID()
Insert into NotificationType
values (@NotificationTypeId , 1, 20, @NotificationDeliveryTypeId,
'DONOTREPLY@aptean.com',
'ProductProfile.xml', 'sa', getUTCdate(),'sa',getUTCdate(), 1, 'http://localhost:55000/20',
'UI:OEP:Notifications')
select @NotificationTypeId = NotificationTypeId
from NotificationType
where SecondaryId=20
Insert into NotificationTypeMl
Values(@NotificationTypeId, 1, 'ENG', 'Any incident assigned to me', 'You will be notified when
an incident is assigned to you.')
```

Associate events to notification type

After creating the notification type, the notification has to be associated with specific events like Incident, Address, Email Address etc., Each notification can be associated with single or multiple events. This association will be done in "NotificationEvent" table.

Column Name	Description
NotificationEventId	Enter a unique GUID for the notification event.
SiteId	Enter the Onyx site ID.
SecondaryId	Enter the Secondary ID for the notification. This is a unique value for each row in the table.
NotificationTypeId	Enter the foreign key of the NotificationType table.
EventId	Enter the foreign key of the NotificationType table.

Adding change fields to notification event

To optimize notifications, we use NotificationEventChange table in persistence db. For example, if there is a notification “Incident assigned to me is Resolved”, and the status is changed to "Resolved" in OEP, then only ONS will process the request.

Column Name	Description
NotificationEventChangeId	Enter a ChangeId for the notification event.
SiteId	Enter the Onyx site ID.
SecondaryId	Enter the Secondary ID for the notification. This is a unique value for each row in the table.
NotificationTypeId	Enter the foreign key of the NotificationType table.
PropertyName	Enter the property name of the object. For ex. StatusId in incident.

Configuring Notification Processes

Before you can begin using the Notifications feature, you must install EMF in your environment.

Mapping the database to the EMF server

Before configuring EMF to work with Onyx, configure Onyx to be able to connect with the EMF server.

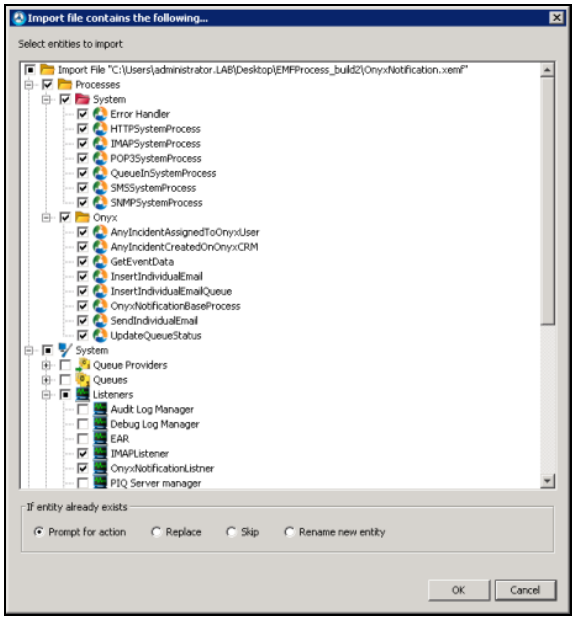
1. In the Persistence database, open the NotificationType table.
2. In the brokerURL column for each row in the table, type the EMF server name and port number.

Importing default processes

1. In your Onyx installation package, navigate to Platform Components>EMF Server>EMFProcess.
2. Copy the OnyxNotification.xemf file to your EMF server.
3. Login to the repository.

   To configure EMF, in the **Audit Log Configuration** dialog uncheck the option **Full logging of all changes** before the import of EMF processes.

4. Right-click on the Root Folder, and click Import.
5. Navigate to the folder where you copied the OnyxNotification.xemf file.
6. Select the OnyxNotification.xemf file, and click Import. A window appears with a check box against each entity that will be imported.



1. Select Prompt for action to be prompted if a selected entity already exists so that you can review which entities get replaced.
2. Click OK to start the import.
3. When the import is complete, the default processes are created in EMF under Root Folder>Onyx.

Configuring Data Source

Map the JDBC data source to your database and OEAS server.

1. Navigate to Data Sources and open the JDBC Data Source window.
2. Enter a Connection Name to identify the Onyx connection. Accept the default value in the Driver/Data Source field.
3. Enter the System Administrator user name and password to connect to the database and the Application server.
4. Under Properties,
   - In the serverName field, enter the name of your database server and SQL server instance.
   - In the serverName field, enter the name of your database server and SQL server instance.
5. Click Apply and close the window.

Configuring SMTP Services

Map the EMF SMTP Services to your Exchange server.

1. Navigate to Services>SMTP Services>OnyxExchangeServer, and open the SMTP Service window.
2. Enter the Name for the SMTP configuration.
3. In the From field, enter a name that will be displayed in the email notification.
4. In the From address field, enter a valid email address that will be used to send email notifications.
5. In the Email Server field, enter the IP address of the email server.
6. Enter the Port number to use to communicate with the email server.
7. Enter the time in seconds after which EMF will stop trying to connect with the email server.
8. Select the authentication type.
9. Enter the User name and password corresponding to the email address that will be used to send email notifications.
10. Click OK to save your changes and close the window.

Configuring Notifications

Before you can begin using the Notifications feature, you must configure Onyx to be able to connect with the email server that you intend to use.

To configure notifications:

1. In OES, enter the system parameter for the OGS path. This is required to communicate with OGS and publish event data.
   - In OES System Parameter Administration, scroll to the parameter GenericPublish.
   - Enter the parameter value in the format `http://[AppServerAddress]:[OGS Port Number]/ServiceGateway/Notification/GenericPublish/`, replacing `AppServerAddress]` with the IP address or name of the app server, and `[OGS Port Number]` with the port number configured for OGS.
   - Save your changes.

Enabling Notifications

The Notifications feature is enabled by default. Use this information to re-enable the feature if you have previously disabled it.

- To enable a previously disabled notification, see Enabling Specific Notifications.
- To disable one or more notifications, see Disabling Specific Notifications.
- To disable the Notifications feature, see Disabling Notifications.

To re-enable Notifications:

1. Show the Notifications tab within the User Preferences screen.
   - On the OEP server, navigate to the **config.xml** file. The default file location is **C:\Program Files\Onyx\EmployeePortal\config.xml**.
   - In an editor such as Notepad, open the **config.xml** file.
   - Search for the node **<tabDef:tab id="Notification**.
   - Set the value of the attribute **visible** to **1**.
   - Save and close the **config.xml** file.

Disabling Notifications

The Notifications feature is enabled by default. You can disable this feature if required, based on your organization's needs.

- To disable one or more notifications, see Disabling Specific Notifications.
- To enable the Notifications feature, see Enabling Notifications.
- To enable a previously disabled notification, see Enabling Specific Notifications.

To disable Notifications:

1. Hide the Notifications tab within the User Preferences screen.

- On the OEP server, navigate to the config.xml file. The default file location is C:\Program Files\Onyx\EmployeePortal\config.xml.
- In an editor such as Notepad, open the config.xml file.
- Search for the node tabDef:tab id="Notification.
- Set the value of the attribute visible to 0.
- Save and close the config.xml file.

Caution: Stop these services only if you intend to disable both, Notifications and Calendar Appointment features.

Caution: Stop these services only if you intend to disable both, Notifications and Calendar Appointment features.

Enabling Specific Notifications

You can enable specific notifications that you previously disabled. When you disable a specific notification, all users who have subscribed to the notification are automatically unsubscribed. The notification is no longer visible in the Available Notifications list in OEP. When you re-enable the notification, users must re-subscribe to receive email alerts.

- To disable a specific notification, see Disabling Specific Notifications.
- To disable the Notifications feature, see Disabling Notifications.
- To enable the Notifications feature, see Enabling Notifications.

1. Restart processing of emails for the notification and make the notification available in the Notification Subscription window in OEP.
   a. In the **NotificationType** table in Persistence database, note the value in the **SecondaryId** column that corresponds to the notification that you want to enable.
   b. On Persistence database, run the following SQL script, replacing [value] with the secondaryId value that you previously noted.

```
/****** Script for SelectTopNRows command from SSMS ******/
declare @NotificationTypeId uniqueidentifier
select @NotificationTypeId = NotificationTypeId
from NotificationType
where secondaryId=[value]
update NotificationType set RecordStatus = 1
where NotificationTypeId = @NotificationTypeId
```

Disabling Specific Notifications

You can disable specific notifications that do not fit your business needs. When you disable a specific notification, all users who have subscribed to the notification are automatically unsubscribed. The notification is no longer visible in the Available Notifications list in OEP. When you re-enable the notification, users must re-subscribe to start receiving email alerts.

- To disable the Notifications feature, see Disabling Notifications.
- To enable the Notifications feature, see Enabling Notifications.
- To enable a previously disabled notification, see Enabling Specific Notifications.

To disable specific notifications:

1. Stop processing of any pending emails for the disabled notification, and make the notification unavailable in the Notification Subscription window in OEP.

   a. In the NotificationType table in Persistence database, note the value in the SecondaryId column that corresponds to the notification that you want to disable.

   b. On Persistence database, run the following SQL script, replacing the [value] with the secondaryId value that you previously noted.

```
/****** Script for SelectTopNRows command from SSMS ******/

declare @NotificationTypeId uniqueidentifier

select @NotificationTypeId = NotificationTypeId

from NotificationType

where secondaryId=[value]

update NotificationType set RecordStatus = 0 where
NotificationTypeId = @NotificationTypeId

Delete from NotificationSubscription where NotificationTypeId =
@NotificationTypeId
```

Creating LBO Adapters Using Step Component

LBO adapter is a .NET 4.5 step component. You can create a new LBO Adapter using the Onyx step component. The LBO Adapter captures events happening on a particular object, collects and publishes eventData to the configured OGS. The LBO Adapter is the Out-of-the-box enabled notification method in the Onyx application.

Default Objects

The following default objects are supported using the LBO Adapter:

- incident
- task
- workNoteDetail

- individual
- company
- address
- phone
- emailAddress

Note: For all the above objects LBO adapter is supported only on Insert, Update, and Savecollection methods. Certain methods like batch update, merge is not supported.

In the following sample of eventData, the LBO Adapter captures the objectName, action (insert/update), oldData (in case of update), newData, and any other context related information.

```
<eventData>
<object>incident</object>
<action>update</action>
<appData>
<oldData>
<incident objectType="incident" content="all">
<primaryId>57F19AAB-DCE6-41B1-AC4A-514A66880FEC</primaryId>
<secondaryId>163</secondaryId>
<ownerId>06B55F12-3705-4308-BA7E-D10570CC82AF</ownerId>
<ownerName>Argyle Fox</ownerName>
<ownerType>2</ownerType>
<contactId null="1"></contactId>
<contactObjectType null="1"></contactObjectType>
<categoryId>1</categoryId>
<typeId>100075</typeId>
<assignedId null="1"></assignedId>
<trackingId>2395</trackingId>
<trackingCode>EUOEP031201Be</trackingCode>
<serialNumber null="1"></serialNumber>
<productId>PPWU400</productId>
<description1>Customer does not want fix it</description1>
<description2 null="1"></description2>
<keyWords>cgbfnfd</keyWords>
```

```
<sourceId>132</sourceId>
<statusId>130</statusId>
<priorityId>127</priorityId>
<code1>101831</code1>
<code2 null="1"></code2>
<code3 null="1"></code3>
<code4 null="1"></code4>
<totalTime>24</totalTime>
<totalLabor>12</totalLabor>
<image>0</image>
<assignedTo>sa</assignedTo>
<insertBy>sa</insertBy>
<insertDate>2002-01-10 21:38:19</insertDate>
<updateBy>sa</updateBy>
<updateDate>2013-08-27 11:10:50</updateDate>
<onyxTimestamp>0000000000017B23</onyxTimestamp>
<privateAccess>0</privateAccess>
<readOnlyAccess>0</readOnlyAccess>
<user1 null="1"></user1>
<user2 null="1"></user2>
<user3>0</user3>
<user4 null="1"></user4>
<user5 null="1"></user5>
<user6 null="1"></user6>
<user7 null="1"></user7>
<user8 null="1"></user8>
<user9 null="1"></user9>
<user10 null="1"></user10>
<locked null="1"></locked>
<lockedBy null="1"></lockedBy>
<alertId null="1"></alertId>
<watch>0</watch>
</incident>
```

```
</oldData>
<newData>
<incident objectType="incident" content="all">
<primaryId>57F19AAB-DCE6-41B1-AC4A-514A66880FEC</primaryId>
<secondaryId>163</secondaryId>
<ownerId>06B55F12-3705-4308-BA7E-D10570CC82AF</ownerId>
<ownerName>Argyle Fox</ownerName>
<ownerType>2</ownerType>
<contactId null="1"></contactId>
<contactObjectType null="1"></contactObjectType>
<categoryId>1</categoryId>
<typeId>100075</typeId>
<assignedId null="1"></assignedId>
<trackingId>2395</trackingId>
<trackingCode>EUOEP031201Be</trackingCode>
<serialNumber null="1"></serialNumber>
<productId>PPWU400</productId>
<description1>Csutomer does not want fix it</description1>
<description2 null="1"></description2>
<keyWords>cccc</keyWords>
<sourceId>132</sourceId>
<statusId>130</statusId>
<priorityId>127</priorityId>
<code1>101831</code1>
<code2 null="1"></code2>
<code3 null="1"></code3>
<code4 null="1"></code4>
<totalTime>27</totalTime>
<totalLabor>15</totalLabor>
<image>0</image>
<assignedTo>sa</assignedTo>
<insertBy>sa</insertBy>
<insertDate>2002-01-10 21:38:19</insertDate>
```

```
<updateBy>sa</updateBy>
<updateDate>2013-08-27 11:14:02</updateDate>
<onyxTimestamp>0000000000017B2A</onyxTimestamp>
<privateAccess>0</privateAccess>
<readOnlyAccess>0</readOnlyAccess>
<user1 null="1"></user1>
<user2 null="1"></user2>
<user3>0</user3>
<user4 null="1"></user4>
<user5 null="1"></user5>
<user6 null="1"></user6>
<user7 null="1"></user7>
<user8 null="1"></user8>
<user9 null="1"></user9>
<user10 null="1"></user10>
<locked null="1"></locked>
<lockedBy null="1"></lockedBy>
<alertId null="1"></alertId>
<watch>0</watch>
</incident>
</newData>
</appData>
<context>
<sessionId>A9B31166-57E8-4C38-9044-AFD56698530D</sessionId>
<userId>sa</userId>
<partnerUser>0</partnerUser>
<siteId>1</siteId>
<applicationName>Onyx</applicationName>
<source>OEP</source>
<preferredLanguage>ENG</preferredLanguage>
<originalUser>sa</originalUser>
<permissionMask>FFFF</permissionMask>
<logName>Onyx</logName>
```

```
<logSettings>1</logSettings>
<impersonatingUser></impersonatingUser>
<sessionSecondsRemaining>600</sessionSecondsRemaining>
<otmId>EEA6CF0B-187B-488C-BE22-203354BC6C82</otmId>
<eventDate>8/27/2013 11:14:02 AM</eventDate>
<eventSource>oeas</eventSource>
</context>
</eventData>
```

Creating a new LBO Adapter

You can create a new LBO Adapter using Onyx Step Component. To do this, perform the following steps:

1. Determine the object and method on which to execute the Step that triggers the notification.
2. Update Persistence database to create entries for the new notification. For information on doing this, see Update database for custom notifications.
3. Create a new step component for the event. Use the sample step component included in your installation package as a guideline for doing this. Ensure that you call the Publish API system parameter from the step component in order to send data to OGS.

   Use the Sample.xml file included in the customization folder for the format of the data XML of the API call. You cannot change the <EventManifest>, <AppData>, and <EventData> elements in the XML file. Within the <EventData> element, add nodes for each property that you want to include.

4. Modify the method to execute the step component in the post event, based on when you want to trigger the notification alert. If you have added other customizations to the method, ensure that these are not affected by this addition.
5. Modify the LBOAdapter.config file on your OEAS server. The default file location is C:\Program Files\Onyx\AppServer\Components\LBOAdapter.config.

You can also use one of the existing templates and extend the LBO Adapters to other objects.

For Normal Objects like incident or task:

1. Copy “firePostInsertEvent” step in “insert” method of “incident” object to “insert” method of other object.
2. Similarly copy “firePreUpdateEvent”, “firePostUpdateEvent” steps in “update” method of “incident” object to update method of other object.

For Collection Objects like address, phone, emailAddress:

1. Copy “firePreChildUpdateEvent” and “firePostChildUpdateEvent” steps in saveCollection method of “address” object to other object.

Modifying the LBO adapter configuration file

After adding a new step component to the method, you must also modify the LBO adapter configuration file to add details for the new notification. Use the following table to understand the values to enter for each attribute.

To modify the LBO adapter configuration file:

1. On your OEAS server, open the LBOAdapter.config file. The default file path is C:\Program Files\Onyx\AppServer\Components\LBOAdapter.config.
2. Use the following table to understand the values to enter for each attribute.

Element	Attribute	Description
Application	name	Enter the name of the logical application that will be used for the notifications.
Application	enable	Enter "true" to enable LBO adapters for the application. Otherwise, enter "false"
Site	id	Enter the site ID that you have configured for the logical application.
Site	enable	Enter "true" to enable LBO adapters for the site. Otherwise, enter "false".
Site	url	Enter the URL to post the data to OGS using the Publish API.
child elements of site: incident/task/workNoteDetail/individual /company/address/phone/emailAddress		Specify the settings for an lbo adapter for the specified site for the specified logical application.
lbo	enable	Set true to use LBO adapters to send notifications for the specified LBO object. Set false to disable LBO adapters for the object.
lbo	retrieveMethod	Enter the name of the LBO retrieve method to use.
retrieveTemplate	parameters	Specify the parameters for the LBO retrieve XML template.
params		Specify how to fill the LBO retrieve template.
param	xpath	Enter the node that will be used in the “retrieveTemplate/parameters”.
param	value	Enter the path of the steppackage.xml file from where we retrieve the value to fill the retrieveTemplate.

Example LBOAdapter configuration file

```
<?xml version="1.0" encoding="utf-8" ?>
<lboAdapter>
  <application name="Onyx" enable="true">
  <site id="1" enable="true"
url="http://localhost:69/ServiceGateway/Notification/GenericPublish/">
  <incident enable="true" retrieveMethod="retrieve">
   <retrieveTemplate>
   <parameters>
    <incident>
     <primaryId></primaryId>
    </incident>
   </parameters>
   </retrieveTemplate>
  <params>
   <param xpath ="incident/primaryId">
    <incident value="incident/primaryId" />
  </param>
  </params>
 </incident>
  <task enable="true" retrieveMethod="retrieve">
   <retrieveTemplate>
    <parameters>
     <task>
      <primaryId></primaryId>
     </task>
    </parameters>
    </retrieveTemplate>
    <params>
    <param xpath ="task/primaryId">
      <task value="task/primaryId" />
    </param>
    </params>
```

```
</task>
<workNoteDetail enable="true" retrieveMethod="retrieve">
 <retrieveTemplate>
  <parameters>
   <workNoteDetail>
    <primaryId></primaryId>
   </workNoteDetail>
  </parameters>
 </retrieveTemplate>
 <params>
  <param xpath ="workNoteDetail/primaryId">
   <workNoteDetail value="workNoteDetail/primaryId" />
 </param>
</params>
</workNoteDetail>
<individual enable="true" retrieveMethod="retrieve">
 <retrieveTemplate>
  <parameters>
   <individual>
    <primaryId></primaryId>
   </individual>
  </parameters>
 </retrieveTemplate>
 <params>
  <param xpath ="individual/primaryId">
   <individual value="individual/primaryId" />
  </param>
 </params>
</individual>
 <company enable="true" retrieveMethod="retrieve">
<retrieveTemplate>
<parameters>
<company>
```

```
<primaryId></primaryId>
</company>
</parameters>
</retrieveTemplate>
<params>
<param xpath ="company/primaryId">
<company value="company/primaryId" />
</param>
</params>
</company>
<address enable="true" retrieveMethod="retrieveCollection">
<retrieveTemplate>
<parameters>
<addresses>
<address>
<ownerId></ownerId>
</address>
</addresses>
</parameters>
</retrieveTemplate>
<params>
<param xpath ="addresses/address/ownerId">
<individual value="individual/primaryId" />
<company value="company/primaryId" />
<address value="addresses/address/ownerId" />
</param>
</params>
</address>
<phone enable="true" retrieveMethod="retrieveCollection">
<retrieveTemplate>
<parameters>
<phones>
<phone>
```

```
<ownerId></ownerId>
</phone>
</phones>
</parameters>
</retrieveTemplate>
<params>
<param xpath ="phones/phone/ownerId">
<individual value="individual/primaryId" />
<company value="company/primaryId" />
<phone value="phones/phone/ownerId" />
</param>
</params>
</phone>
<emailAddress enable="true" retrieveMethod="retrieveCollection">
<retrieveTemplate>
<parameters>
<emailAddresses>
<emailAddress>
<ownerId></ownerId>
</emailAddress>
</emailAddresses>
</parameters>
</retrieveTemplate>
<params>
<param xpath ="emailAddresses/emailAddress/ownerId">
<individual value="individual/primaryId" />
<company value="company/primaryId" />
<emailAddress
value="emailAddresses/emailAddress/ownerId" />
</param>
</params>
</emailAddress>
</site>
```

```
</application>
</lboAdapter>
```

Enabling/Disabling LBO Adapter

An LBO Adapter supports multiple logical applications and multi-site environments. Out of the box, one logical application and one site is included with the Onyx application. The LBO adapter identifies an application configuration by checking the “name” attribute. Similarly, the LBO Adapter identifies the configuration of a particular site by checking the “id” attribute of the “site” Element inside a particular application.

The LBO Adapter can be enabled or disabled for an application, site or object.

Note: The ideal way to disable the LBO Adapter for a single application or a single object is by disabling “firePostInsertEvent” step in insert method and “firePreUpdateEvent”, “firePostUpdateEvent” steps in update method. However, this approach requires an OED publish.

The ideal way to disable the LBO Adapter for a single application or a single object is by disabling “firePostInsertEvent” step in insert method and “firePreUpdateEvent”, “firePostUpdateEvent” steps in update method. However, this approach requires an OED publish.

To Enable/Disable LBO Adapter for a single logical application:

1. Open the LBO Adapter configuration file.
2. Use the application name to search for the specific application configuration.
3. Change the enable property to true/false.

To Enable/Disable LBO Adapter for a single site:

1. Open the LBO Adapter configuration file.
2. Search for the specific logical application Element.
3. Use the site id to search for the specific site.
4. Change the enable property to true/false.

To Enable/Disable LBO Adapter for a single object:

1. Open the LBO Adapter configuration file.
2. Search for the specific logical application Element.
3. Use the site id to search for the specific site.
4. Change the enable property to true/false.

To configure the OGS URL:

1. Open the LBO Adapter configuration file.
2. Search for the specific logical application Element.

3. Use the site id to search for the specific site.
4. Configure the “url” property.

Adding UI Resource Permission

Each notification on the notification screen is secured using a UI:Resource. This UI resource can be configured and user permissions can be set using the security administration present in OEAS.

The default UI: resource for all the notifications is UI:OEP:Notifications.

ONS Extensions

The Onyx application processes and removes all unwanted records from the LBO Adapter/SQL Adapter and this information is stored in the database.

By default, the Onyx application provides customers with the following pre-process actions.

- Remove Embedded XML
- Remove Child Objects
- Add ActionType Attribute

However, customers can write their own extensions or remove existing actions. The path for the configurations is **C:\Program Files\Onyx\AppServer\Applications\Onyx\OnyxNotificationService\Config\adapterExtension.config file**.

Remove Embedded XML

In individual and company objects, an embedded xml is found in the CData component. CData contains information of other objects like company, phone etc., which is not required for an individual or company object and can be removed. This is applicable to other objects as well.

The out-of-the-box action to remove an embedded xml is:

```
<action id="1" name="Remove Embedded Xml"
interface="Onyx.ONS.AdapterExtension.Base.ICData" />
```

Remove Child Objects

In Individual and Company objects, the adapter sends information of the dependent objects along with the main object. To eliminate these dependent objects we can use the Remove Child Objects action.

The out-of-the-box action to Remove a Child Object is:

```
<action id="2" name="Remove Child Objects"
interface="Onyx.ONS.AdapterExtension.Base.IChildObject" >
<remove>
<phones/>
<addresses/>
```

```
<emailAddresses />
</remove>
</action>
```

Add ActionType Attribute

In collections objects like address, emailAddress, or phones, an action is sent as “saveCollection”. This could also be a customized action name from the LBO adapter. To update the action type, change the action name in “ oldValueLBO” and newValueLbo and select “update”.

The out-of-the-box action to Add an Action Type is:

```
<action id="2" name="Add ActionType Attribute"
interface="Onyx.ONS.AdapterExtension.Base.ICollectionObject" >
```

Filtering Actions

Along with the pre-process actions, the Onyx application provides the following filtering actions:

- Discard Empty Data Event
- Discard Invalid Event
  - Ignore
  - ignoreWithCondition

Discard Empty Data Event

NewData and OldData nodes may contain only object names. For example, if only individual details (email id of the individual) is updated without an address in the record, then the address object will be reflected as empty data.

The out-of-the-box action to discard empty data is:

```
<action id="3" name="Discard Empty Data Event"
interface="Onyx.ONS.AdapterExtension.Base.IEmptyEvent" />
```

Discard Invalid Event

Only certain entities in an object like firstname, salutation etc., can be removed from an object.

The out-of-the-box action to discard an Invalid Event is:

```
<action id="4" name="Discard Invalid Event"
interface="Onyx.ONS.AdapterExtension.Base.IInvalidEvent" >
<ignore>
<updateBy />
<updateDate />
<onyxTimestamp />
</ignore>
<ignoreWithCondition>
```

```
<conditions>
<primary newValueSql="True" newValueLbo="1" />
</conditions>
<ignore>
<firstName />
<middleName/>
<lastName/>
<salutation/>
<suffix />
<companyName/>
</ignore>
</ignoreWithCondition>
<ignoreWithCondition>
<conditions>
<deleteStatus newValueSql="2" newValueLbo="2" />
</conditions>
<ignore>
<deleteStatus />
</ignore>
</ignoreWithCondition>
<ignoreWithCondition>
<conditions>
<primary oldValueSql="True" newValueSql="False" oldValueLbo="1"
newValueLbo="0"/>
</conditions>
<ignore>
<primary />
</ignore>
</ignoreWithCondition>
</action>
```

The Discard Invalid Event is divided into:

Ignore

This action will ignore all child nodes. For example, a change in the event “updateDate” inside an ignore node, will not be updated as the system will ignore the event.

The out-of-the-box action looks for Ignore is:

```
<ignore>
<updateBy />
<updateDate />
<onyxTimestamp />
</ignore>
```

ignoreWithCondition

If a node has an entity name and the conditions oldValueLbo and newValueLbo are similar, then that particular eventData will be ignored.

The out-of-the-box action for IgnoreWithCondition is:

```
<ignoreWithCondition>
<conditions>
<primary oldValueSql="True" newValueSql="False" oldValueLbo="1"
newValueLbo="0"/>
</conditions>
<ignore>
<primary />
</ignore>
</ignoreWithCondition>
```

The following is the ONS Extension Config file

```
<?xml version="1.0" encoding="utf-8" ?>
<adapterExtension>
<reference assembly="Onyx.ONS.AdapterExtension.Base.dll" />
<IEventDataFilter>
<individual enable="true" primaryKey="primaryId"
class="Onyx.ONS.AdapterExtension.Base.IndividualFilter,
Onyx.ONS.AdapterExtension.Base, PublicKeyToken=206c397fab0ffa11">
<action id="1" name="Remove Embedded Xml"
interface="Onyx.ONS.AdapterExtension.Base.ICData" />
<action id="2" name="Remove Child Objects"
interface="Onyx.ONS.AdapterExtension.Base.IChildObject" >
```

```
<remove>
<phones/>
<addresses/>
<emailAddresses />
</remove>
</action>
<action id="3" name="Discard Invalid Events"
interface="Onyx.ONS.AdapterExtension.Base.IInvalidEvent" >
<ignore>
<updateBy />
<updateDate />
<onyxTimestamp />
</ignore>
</action>
</individual>
<company enable="true" primaryKey="primaryId"
class="Onyx.ONS.AdapterExtension.Base.CompanyFilter,
Onyx.ONS.AdapterExtension.Base, PublicKeyToken=206c397fab0ffa11">
<action id="1" name="Remove Embedded Xml"
interface="Onyx.ONS.AdapterExtension.Base.ICData" />
<action id="2" name="Remove Child Objects"
interface="Onyx.ONS.AdapterExtension.Base.IChildObject" >
<remove>
<phones/>
<addresses/>
<emailAddresses />
</remove>
</action>
<action id="3" name="Discard Invalid Event"
interface="Onyx.ONS.AdapterExtension.Base.IInvalidEvent" >
<ignore>
<updateBy />
<updateDate />
<onyxTimestamp />
</ignore>
```

```
</action>
</company>
<address collection="1" primaryKey="primaryId" enable="true"
class="Onyx.ONS.AdapterExtension.Base.AddressFilter,
Onyx.ONS.AdapterExtension.Base, PublicKeyToken=206c397fab0ffa11">
<action id="1" name="Remove Embedded Xml"
interface="Onyx.ONS.AdapterExtension.Base.ICData"/>
<action id="2" name="Add ActionType Attribute"
interface="Onyx.ONS.AdapterExtension.Base.ICollectionObject" >
<changeType oldValueLbo="saveCollection" newValueLbo= "update" />
</action>
<action id="3" name="Discard Empty Data Event"
interface="Onyx.ONS.AdapterExtension.Base.IEmptyEvent" />
<action id="4" name="Discard Invalid Event"
interface="Onyx.ONS.AdapterExtension.Base.IInvalidEvent" >
<ignore>
<updateBy />
<updateDate />
<onyxTimestamp />
</ignore>
<ignoreWithCondition>
<conditions>
<primary newValueSql="True" newValueLbo="1" />
</conditions>
<ignore>
<firstName />
<middleName/>
<lastName/>
<salutation/>
<suffix />
<companyName/>
</ignore>
</ignoreWithCondition>
<ignoreWithCondition>
<conditions>
```

```
<deleteStatus newValueSql="2" newValueLbo="2" />
</conditions>
<ignore>
<deleteStatus />
</ignore>
</ignoreWithCondition>
<ignoreWithCondition>
<conditions>
<primary oldValueSql="True" newValueSql="False" oldValueLbo="1"
newValueLbo="0"/>
</conditions>
<ignore>
<primary />
</ignore>
</ignoreWithCondition>
</action>
</address>
<phone collection="1" primaryKey="primaryId" enable="true"
class="Onyx.ONS.AdapterExtension.Base.PhoneFilter,
Onyx.ONS.AdapterExtension.Base, PublicKeyToken=206c397fab0ffa11">
<action id="1" name="Remove Embedded Xml"
interface="Onyx.ONS.AdapterExtension.Base.ICData" />
<action id="2" name="Add ActionType Attribute"
interface="Onyx.ONS.AdapterExtension.Base.ICollectionObject" >
<changeType oldValueLbo="saveCollection" newValueLbo= "update" />
</action>
<action id="3" name="Discard Empty Data Event"
interface="Onyx.ONS.AdapterExtension.Base.IEmptyEvent" />
<action id="4" name="Discard Invalid Event"
interface="Onyx.ONS.AdapterExtension.Base.IInvalidEvent" >
<ignore>
<updateBy />
<updateDate />
<onyxTimestamp />
</ignore>
```

```
<ignoreWithCondition>
<conditions>
<deleteStatus newValueSql="2" newValueLbo="2" />
</conditions>
<ignore>
<deleteStatus />
</ignore>
</ignoreWithCondition>
<ignoreWithCondition>
<conditions>
<primary oldValueSql="True" newValueSql="False" oldValueLbo="1"
newValueLbo="0"/>
</conditions>
<ignore>
<primary />
</ignore>
</ignoreWithCondition>
</action>
</phone>
<emailAddress collection="1" enable="true" primaryKey="primaryId"
class="Onyx.ONS.AdapterExtension.Base.EmailAddressFilter,
Onyx.ONS.AdapterExtension.Base, PublicKeyToken=206c397fab0ffa11">
<action id="1" name="Remove Embedded Xml"
interface="Onyx.ONS.AdapterExtension.Base.ICData" />
<action id="2" name="Add ActionType Attribute"
interface="Onyx.ONS.AdapterExtension.Base.ICollectionObject" >
<changeType oldValueLbo="saveCollection" newValueLbo= "update" />
</action>
<action id="3" name="Discard Empty Data Event"
interface="Onyx.ONS.AdapterExtension.Base.IEmptyEvent" />
<action id="4" name="Discard Invalid Event"
interface="Onyx.ONS.AdapterExtension.Base.IInvalidEvent" >
<ignore>
<updateBy />
<updateDate />
```

```
<onyxTimestamp />
</ignore>
<ignoreWithCondition>
<conditions>
<deleteStatus newValueSql="2" newValueLbo="2" />
</conditions>
<ignore>
<deleteStatus />
</ignore>
</ignoreWithCondition>
<ignoreWithCondition>
<conditions>
<primary oldValueSql="True" newValueSql="False" oldValueLbo="1"
newValueLbo="0"/>
</conditions>
<ignore>
<primary />
</ignore>
</ignoreWithCondition>
</action>
</emailAddress>
<incident enable="true" primaryKey="primaryId"
class="Onyx.ONS.AdapterExtension.Base.IncidentFilter,
Onyx.ONS.AdapterExtension.Base, PublicKeyToken=206c397fab0ffa11">
<action id="1" name="Discard Invalid Event"
interface="Onyx.ONS.AdapterExtension.Base.IInvalidEvent" >
<ignore>
<updateBy />
<updateDate />
<onyxTimestamp />
</ignore>
</action>
</incident>
```

```
<task enable="true" primaryKey="primaryId"
class="Onyx.ONS.AdapterExtension.Base.TaskFilter,
Onyx.ONS.AdapterExtension.Base, PublicKeyToken=206c397fab0ffa11">
<action id="1" name="Discard Invalid Event"
interface="Onyx.ONS.AdapterExtension.Base.IInvalidEvent" >
<ignore>
<updateBy />
<updateDate />
<onyxTimestamp />
</ignore>
</action>
</task>
<workNoteDetail enable="true" primaryKey="primaryId"
class="Onyx.ONS.AdapterExtension.Base.WorkNoteDetailFilter,
Onyx.ONS.AdapterExtension.Base, PublicKeyToken=206c397fab0ffa11">
<action id="1" name="Discard Invalid Event"
interface="Onyx.ONS.AdapterExtension.Base.IInvalidEvent" >
<ignore>
<updateBy />
<updateDate />
<onyxTimestamp />
</ignore>
</action>
</workNoteDetail>
</IEventDataFilter>
</adapterExtension>
```

Note: Any changes in adapterExtension.config file, restart ONS service.

Default Notifications

The following table lists the default notifications configured in Onyx, along with the corresponding objects.

Notification Name	OTDB Table Name
Any incident assigned to me	incident
Any incident assigned to me is modified	incident
Any incident I created is resolved	incident
Any incident I created has a new work note	work_note_detail
Any details of my favorite customer is modified	Phone, email_address, address, individual, company
Any incident I created is modified	incident
Any incident created in Onyx CRM	incident
Any incident assigned to Onyx user	incident
Incident I am watching has a change	incident
Incident I am watching has a status of "close"	incident
Customer I am watching has any change of address	address
Customer I am watching has a new incident of type "Service" and status "Open“	incident
My favorite customer change of address	address
Any task is assigned to me or any task assigned to me is modified	task

Calendar Appointments

The Calendar Appointments feature uses the recall date and time specified on incident and task records to create an appointment in each identified user's calendar. Users can also view this appointment on the Appointments tab of the associated incident and customer record.

To use Calendar Appointments, you should be using Microsoft Exchange Server 2007 SP1 and higher, or Microsoft Exchange Server 2010. You must also configure Onyx to be able to connect with the Exchange server that you intend to use.

The tasks explained in this topic are:

Configuring Calendar Appointments

In order to use the Calendar Appointments feature along with your corporate Exchange Server, you must have the following information ready before you run the OEAS setup.

- The Exchange Server domain name and URI.
- The impersonation user ID and password to use for updating appointments.

In OES, enter the system parameter for the OGS path. This is required for the SQL CLR trigger to communicate with OGS and publish event data.

- In OES System Parameter Administration, scroll to the parameter **NotificationWebService**.
- Enter the parameter value in the format http://[AppServerAddress]:[OGS Port Number]/ServiceGateway/Notification/Publish/, replacing [AppServerAddress] with the IP address or name of the app server, and [OGS Port Number] with the port number configured for OGS.
- Save your changes.

Permissions for the impersonation user ID

The impersonation user account must be given specific permissions on the Exchange server in order to be able to create appointments for other users.

- On Exchange server 2007, grant permission for the ms-Exch-EPI-Impersonation and ms-Exch-EPI-May-Impersonate roles.
- On Exchange server 2010, grant permission for the ApplicationImpersonation role.

Example settings in the Appointments.config file

When you run the OEAS installer, the Appointments.config file is generated which contains all the information that you entered during installation. You can manually modify this information if required. The default path for the config file is C:\Program Files\Onyx\AppServer\Applications\Onyx\OnyxNotificationService\Config.

The following is an example of the settings in the Appointments.config file.

```
<OutlookServers>

<OutlookServer

ID="1"

EWS="https://10.101.64.176/ews/exchange.asmx"

UserName="serviceaccount"

Password="BeyejpekPjlFLOlOJdv3qOaK3LiYIZH/laJq+6zX0hXCY5pOknliLXHEf6oVwF43YeA/JvGfmpbw8VVgvO6R9+nl8wVSaNBl2

qKOCddVkcunjEuenK+6LYbEVYr5d7NOFWyQtDPX2PBLl1jAl+Lr0qFIEdmIS+ZO/sDeOc8UriM="

EnableSsl="true"

DefaultCredentials="false"

Domain = "lab"

EnableEwsTraceLog ="false"

/>

</OutlookServers>
```

The EnableEwsTraceLog option is set to False by default. We do not recommend turning this option to True as it can adversely affect performance. Turn this option to True only when you need to debug this feature.

Scheduling Calendar Appointments

You can define schedules for the delivery of calendar appointments to your users to optimize resource utilization and suit the demands of your business.

To schedule calendar appointments:

1. On the OEAS server, stop the Onyx.ONS service.
2. Go to the Onyx Notification Service installation path, and open the Schedules.config file. The default file location is C:\Program Files\Onyx\AppServer\Applications\Onyx\OnyxNotificationService\Config\Schedules.config.
3. Using an editor such as Notepad, create schedules to match your business needs. For more information, see Example code for schedules.

The following table provides a description of each property.

Property	Description
Schedule ID	Enter a unique identification number for the schedule you want to set. You can set multiple schedules for each schedule type. Once saved, do not change this value.
Type	Enter the type of schedule you want to create. To create a schedule for calendar appointments, enter CALENDAR_SYNC as the type. Once saved, do not change this value. You can create multiple schedules for a single schedule type.
Enable	Enter true to enable the schedule. Enter false if you do not plan to enable the schedule immediately. When this value is set to false, no emails or appointments are sent during the schedule time.
StartTime	Enter the time, in 24-hour format, at which the scheduler service will begin processing notification emails and calendar appointments. We recommend a minimum gap of 5 minutes between two schedules of the same type. Note that all times are in UTC timezone.
EndTime	Enter the time, in 24-hour format, at which the scheduler service will stop processing notification emails and calendar appointments. To set a schedule that runs 24 hours a day, enter "" as the value for the EndTime attribute. Note that all times are in UTC timezone.

4. After creating the schedules you need, save and close the Schedules file.
5. Restart the Onyx.ONS service.

Example XML for schedules

Example code for a single schedule

```
<? xml version="1.0" encoding="utf-8"?>
<Schedules>
<Schedule ID="1" Type="CALENDAR_SYNC" Enable="true" StartTime="0000" EndTime="0500" />
</Schedules>
```

Example code for multiple schedules

```
<? xml version="1.0" encoding="utf-8" ?>
<Schedules>
<Schedule ID="1" Type="CALENDAR_SYNC" Enable="true" StartTime="0900" EndTime="1800" />
<Schedule ID="2" Type="CALENDAR_SYNC" Enable="true" StartTime="1830" EndTime="0830" /> <!—This is the second schedule for CALENDAR_SYNC type)
</Schedules>
```

Enabling Calendar Appointments

The Calendar Appointments feature is enabled by default. Use this information to re-enable the feature if you have previously disabled it.

To disable the Calendar Appointments feature, see Disabling Calendar Appointments.

To enable Calendar Appointments:

1. Show the 'Send to Calendar' control on the Incident and Tasks pages using UCW.
2. On the database server, run the following script on your OTDB to enable the triggers created for the Calendar Appointments feature, replacing [**Trigger Name**] with the name of each trigger, and [**Table Name**] with the name of the corresponding table.

   ```
   enable TRIGGER [Trigger Name] on [Table Name]
   ```

   Example script:

   ```
   enable TRIGGER otCalendarAppointmentInsert on Reminder
   enable TRIGGER otCalendarAppointmentUpdate on Reminder
   ```

3. On the OEAS server, start/restart the Notifications services if you had previously stopped them.
   - Go to **Control Panel>Administrative Tools> Services**.
   - Start/restart the ONS service. The names of the services are in the format **[Onyx Application Name].ONS**.

Disabling Calendar Appointments

The Calendar Appointments feature is enabled by default. You can disable this feature if required, based on your organization's needs.

To enable the Calendar Appointments feature, see Enabling Calendar Appointments.

To disable Calendar Appointments:

1. Hide the 'Send to Calendar' control on the Incident and Tasks pages using UCW.
2. On the database server, run the following script on your OTDB to disable the triggers created for the Calendar Appointments feature, replacing [Trigger Name] with the name of each trigger, and Table Name with the name of the corresponding table. For a list of trigger names, see SQL Triggers for Appointments.

   ```
   disable TRIGGER [Trigger Name] on [Table Name]
   ```

   Example script:

   ```
   disable TRIGGER otCalendarAppointmentInsert on Reminder
   disable TRIGGER otCalendarAppointmentUpdate on Reminder
   ```

3. On the OEAS server, stop/restart the Notifications services.
   - Go to Control Panel>Administrative Tools> Services.
   - Stop/Restart the ONS and service. The names of the services are in the format [Onyx Application Name].ONS.
4. Stop these services only if you intend to disable both, Notifications and Calendar Appointment features.

Configure SQL Adapter

The SQL triggers are not installed out-of-the-box in the Onyx application and need to be configured and enabled if required. For more information on enabling the SQL Triggers, see Enable / Disable SQL Trigger.

To configure the SQL Adapter:

1. Run utilities\DBASetupScript.sql on the master database and provide input parameter as follows:
   a. SET @onyxDBO = N'onyxDBO'
   b. SET @Pdatabase = N'PersistenceDB71demo'
   c. SET @Odatabase = N'OEDB71demo'

Where **@onyxDBO** is the OnyxDBO user name in database server, **@Pdatabase** is the name of persistence DB and **@Odatabase** is the name of the Onyx Enterprise Database.

2. If the database server is 64-bit, then run the utilities\DbaSetupScriptOEDBx64.sql file on OEDB else, if database server is 32-bit, then run the utilities\DbaSetupScriptOEDBx86.sql on OEDB.
3. Copy all DLL files from Customization Support\Database Server\CLR Trigger\CLRDlls to C:\Program Files (x86)\Onyx\Notification_CLR in case of 32-bit and C:\Program Files \Onyx\Notification_CLR in case of 64-bit.
4. Edit CreateCLRAssmblyTrigger.SQL to find all $(path) lines and replace with C:\Program Files (x86)\Onyx\Notification_CLR in case of 32-bit and C:\Program Files \Onyx\Notification_CLR in case of 64-BIT.

   Example:

   a. CREATE ASSEMBLY oaOnyxCLRTrigger
   b. AUTHORIZATION [dbo]
   c. From '$(path)\oaOnyxCLRTrigger.dll'
   d. WITH PERMISSION_SET = UNSAFE

   In case of 64-bit

   a. CREATE ASSEMBLY oaOnyxCLRTrigger
   b. AUTHORIZATION [dbo]
   c. From ' C:\Program Files (x86)\Onyx\Notification_CLR\oaOnyxCLRTrigger.dll'
   d. WITH PERMISSION_SET = UNSAFE

5. After editing the file, run the file in Onyx Enterprise DataBase.

SQL Triggers for Appointments

Onyx uses two SQL CLR triggers to create and update appointments in users' calendars. The following table lists the SQL CLR triggers used for creating, updating and deleting appointments, along with their corresponding DLL.

Type of operation	DLL Name	SQL CLR Trigger Name	OTDB Table Name
Creating a new appointment for Incidents and Tasks	oaCalendarAppointment.dll	otCalendarAppointmentInsert	Reminder
Updating or deleting an existing appointment for Incidents and Tasks	oaCalendarApointment.dll	otCalendarApointmentUpdate	Reminder

Adding HTML Template File

An HTML template file specifies the captions for each field that is included in the notification email that users receive. It also defines the style and format of the email. You can add a new HTML template file for your custom notification if the default files do not meet your requirements.

For information on the default HTML template files available in your Onyx installation, see About template files.

To create an HTML template file:

1. Open a new HTML file.
2. Using one of the default HTML template files as an example, define the styles to use in the template. Use internal styling; do not reference an external CSS file.
3. Using one of the default HTML template files as an example, create one row for each field that you want to include in your email notification.
   - <td class="style1">: Enter the caption for the field in the first column.
   - <td class="style2">: Enter the template key corresponding to the field from the profile XML file.

   Repeat this process for each field you want to add.
4. Save and close the HTML template file.

   The following is an example of a table row as it appears in the HTML template file.

```
<tr style=p>
<td class="style1">
<b>Type</b> :
</td>
<td class="style2">
##product_desc##
</td>
</tr>
```

Modifying HTML Template Files

Onyx uses metadata stored in Profile XML and HTML template files to control the information that is displayed in calendar appointments. Based on the organization's requirement, these files can be modified to add and remove fields that are displayed to users.

Adding a new field

To add a new field to the template body, you must modify the Profile XML file and the corresponding Template file.

To add a new field:

1. Modify the Profile XML file corresponding to the appointment to enable the new field. To do this, see Modifying the Profile XML file.
2. Open the template file that corresponds to the notification that you want to modify. The default file path is **C:\Program Files\Onyx\AppServer\Applications\Onyx\OnyxNotificationService\Templates**.
3. Add a row for the new field that you want to add, in the following format, replacing the highlighted text in the second column with the template key that you noted while modifying the Profile XML file.

```
<tr style=p>
<td class="style1">
<b>Type</b> :
</td>
<td class="style2">
##incident_type_desc##
</td>
</tr>
```

4. If required, change the styles used in the template body using internal or inline styles. Do not reference an external CSS to modify styles.
5. Save and close the Template file.

Note: To add a field to the Individual template subject, you must also modify the emailtemplates.config file. For information on doing this, see Appointment templates.

Modifying Template Configuration File

Use the Templates.config file to specify the XML and template files that are associated with each calendar appointment. You can define the text that will appear in the subject of calendar appointments. You can also specify whether the appointment will be sent as HTML or plain text.

To modify the fields included in the template, see Modifying Profile XML File and Modify HTML Template Files.

The Template.Config file is located on the OEAS server. The default file path is C:\Program Files\Onyx\AppServer\Applications\Onyx\OnyxNotificationService\Config.

The following table describes the attributes for each element in the Templates.config file. Modify the attribute values as needed to change the information that is displayed in emails and appointments.

Attribute	Description
Type	Do not change this value for existing elements. This attribute specifies that the event is a calendar appointment for Incidents and Tasks.
ID	Do not change this value for existing elements. This attribute represents the calendar appointment that the element is associated with.
HTMLBody	Enter True to send the appointment in HTML format. Enter False to send the appointment in Plain Text format.
ProfileTemplate	This attribute represents the name of the Profile XML file that corresponds to the specified calendar appointment. This value must match the value in the ProfileFileName column of the NotificationType table in the Persistence database corresponding to the appointment.
IndividualSubjectTemplate	This attribute represents the text that appears in the subject field for appointments. From the Profile XML file, enter the TemplateKey value of each field where the IsSubjectTag attribute is set to True. Use the format "[##title##] - ##incident_id## assigned by ##action_user##" to add multiple fields as variables in the subject and connect them with plain text to make a meaningful subject for calendar appointments.
IndividualBodyTemplateFile	This attribute represents the path and name of the template file that is used to populate the body of calendar appointments. Enter the template file name corresponding to the specified appointment in the format Templates\[FileName], where [FileName] is the name of the template file.

Adding Profile XML File

A Profile XML file is a collection of nodes that determine which fields appear in the notification email that users receive. You can add a new profile XML file for your custom notification if the default profile XML files do not meet your requirements.

For information on the default profile XML files available in your Onyx installation, see About template files.

To create a profile XML file:

1. Open a new XML file.
2. Using one of the default profile XML files as an example, create one node for each field that you want to include in your email notification.

- Element name: Enter the LBO property name for the field you want to include.
- IsSubjectTag: Set this attribute to 'true' if you want the field to appear in the email subject. Otherwise enter 'false'.
- IsBodyTag: Set this attribute to 'true' if you want the field to appear in the email body. Otherwise enter 'false'.
- TemplateKey: Enter a value in the format ##[text]##. This value must be entered in the HTML template file for the field to display.

The following is an example of a node as it appears in the Profile XML file.

```
<Description
IsSubjectTag="true"
IsBodyTag="true"
TemplateKey="##desc1##">
</Description>
```

3. Create a node named action. This represents the action on the record that triggers the notification. For example, whether the notification is triggered when a record is inserted, updated, or deleted.
4. Create a node named actionUser. This represents the user performing the action that triggers the notification.

   The following is an example of an action and an actionuser node:

   ```
   <action IsSubjectTag="true" IsBodyTag="true"
   TemplateKey="##action##"></action>
   <actionUser IsSubjectTag="true" IsBodyTag="true"
   TemplateKey="##action_user##"></actionUser>
   ```

5. After creating nodes for each required field, save the profile XML file in the installed path folder. For example, c:\Program Files\Onyx\AppServer\Applications\Onyx\OnyxNotificationService\Profiles folder with an appropriate name.

Modifying Profile XML File

Onyx uses metadata stored in Profile XML and HTML template files to control the information that is displayed in calendar appointments. Based on the organization's requirement, these files can be modified to add and remove fields that are displayed to users.

To add a field to the template:

1. On your OEAS server, open the Profile XML file that you want to modify. The default file path is C:\Program Files\Onyx\AppServer\Applications\Onyx\OnyxNotificationService\Profiles.

The fields that are not currently a part of the template are commented out. The field name is the same as that of the corresponding LBO property.

2. To include a field in the template, uncomment it in the Profile XML.
   - To include the field in the template subject, set the IsSubjectTag attribute to True.
   - To include a field in the template body, set the IsBodyTag attribute to True.
   - Note the value of the TemplateKey attribute which you will need to use later.
3. Save and close the Profile XML file.
4. Modify the HTML template file corresponding to the appointment. To do this, see Modifying HTML Template Files.

> **Note:** To add a field to the Individual template subject, you must also modify the emailtemplates.config file. For information on doing this, see Modifying Template Configuration File.

Viewing Appointment Logs

You can view the event log for calendar appointments in Windows Event Viewer. SQL CLR trigger events are logged on your database server while ONS events are logged on the application server.

SQL CLR events on the database server can be identified by Source = Onyx.CLR and Event ID = 243.

Logs for calendar appointments are stored in the Persistence database. You can use this information to view the status of an appointment, as well as to view and resolve errors. You can also export this information for auditing and security checks.

The following tables in the Persistence database store logs of notification emails and calendar appointments that are sent to users.

Feature	Table Name in Persistence database
Calendar Appointments	OutlookPublishEvent

Use the following table to understand the status of each calendar event.

Status	Description
0	The calendar event is yet to be processed.
1	The calendar event was processed successfully.
2	The calendar event was processed but with errors. Use the ExceptionDetails column to view the errors. After resolving the errors, change the status to 0 for re-processing.
4	This error occurs when the calendar appointment is not created because the exchange server address or the email ID configured is not valid.

Status	Description
	Verify that you have configured a valid exchange server, and valid email IDs for each Onyx user. After resolving the errors, change the status to 0 for re-processing. In this status, if any new appointments (for valid email IDs) were created successfully, these are rolled back. However, any appointments that were updated or deleted are not rolled back.

Modifying Appointment Templates

Based on your organization's requirement, you can control the information that is sent to users in their calendar appointments. Use the Templates.config file to define the subject and body of each calendar appointment that is sent to Onyx users. For each calendar appointment, you can specify the tables, XML, and HTML files from which data is populated.

For a list of profile XML and HTML template files for appointments, see Template files.

To add fields to templates, you must modify the following files:

- Template Config file
- Profile XML files
- HTML Template files

To remove fields from templates, see Removing fields from templates.

Template Files

Onyx uses profile XML and HTML template files to control the information that users receive in their calendar appointments. These files are located within the Onyx installation folder on the OEAS server.

The default path for profile XML files is: C:\Program Files\Onyx\AppServer\Applications\Onyx\OnyxNotificationService\Profiles.

The default path for HTML template files is: C:\Program Files\Onyx\AppServer\Applications\Onyx\OnyxNotificationService\Templates.

You can configure Onyx to send appointment notifications in plain text format. The default plain text templates are included in your installation package under Customization Support>Application Server (OEAS)>OnyxNotificationService>PlainTextTemplates. For information about configuring Onyx to use plain text templates, see Modifying the template configuration file.

The following table lists the files associated with each appointment type.

Type	Name	Profile XML file	HTML template file
Appointment	Appointment for incidents	CalendarAppointmentIncidentProfile.xml	CalendarAppointmentIncident.html
Appointmen	Appointmen	CalendarAppointmentTaskProfile.xml	CalendarAppointmentIncident.ht

Type	Name	Profile XML file	HTML template file
t	t for tasks		ml

Removing Fields from Templates

To remove a field from the appointment template:

1. Open the Profile XML file corresponding to the appointment.
2. Comment out the field, and save the file.
3. Open the HTML Template file corresponding to the appointment.
4. Remove or comment the row corresponding to the field from the HTML Template file, and save the file.
5. If the field you removed is in the template subject line, open the Templates.config file and find the element corresponding to the appointment you want to modify.
6. From the IndividualSubjectTemplate attribute, remove the TemplateKey that corresponds to the field you want to remove.
7. Ensure that the subject text is meaningful after removing the field.
8. Save the Templates.config file.

Event Management Framework

Event Management Framework (EMF) application is a tool that can monitor business data and notify selected recipients on significant events as and when it occurs. EMF has a workflow designer which is very easy to customize and provides a flexible way to react to events that are generated from Onyx CRM.

EMF Architecture

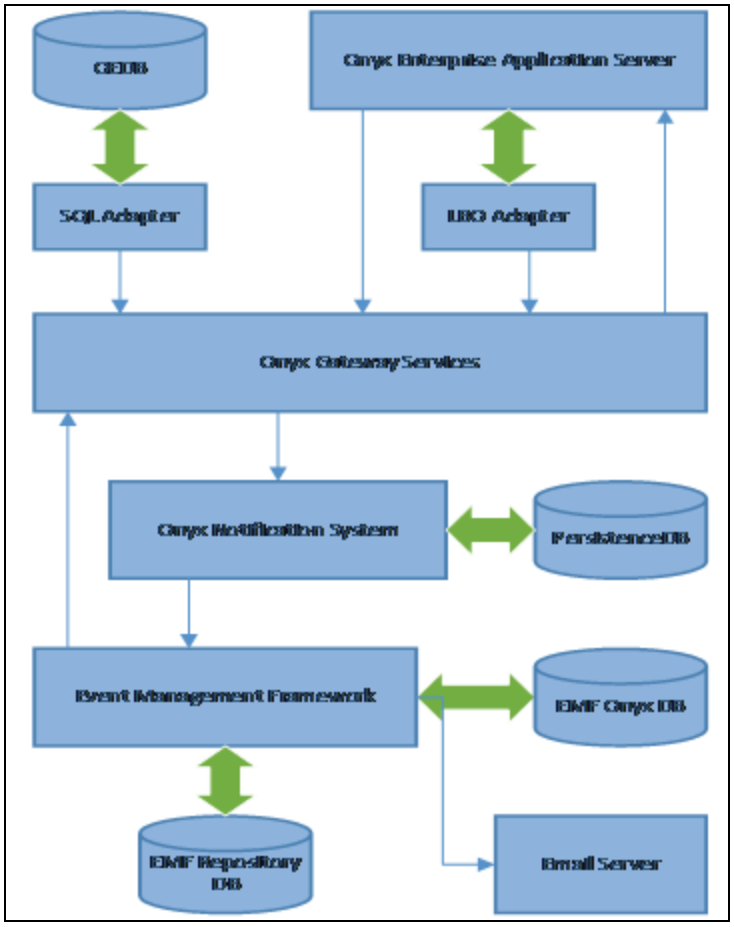
This section describes the software requirement and objective that have significant impact on the architecture.

Onyx Platform

Any change in data on the Onyx platform will initiate a request to EMF. The events are captured by LBO Adapters and notified to EMF.

EMF Platform

Onyx to EMF event notification occurs on http protocol. To support this, EMF has http listeners which listen to events that are generated from Onyx CRM. Interaction from EMF to Onyx occurs on http protocol and always uses onyx gateway services.



Onyx-EMF Integration

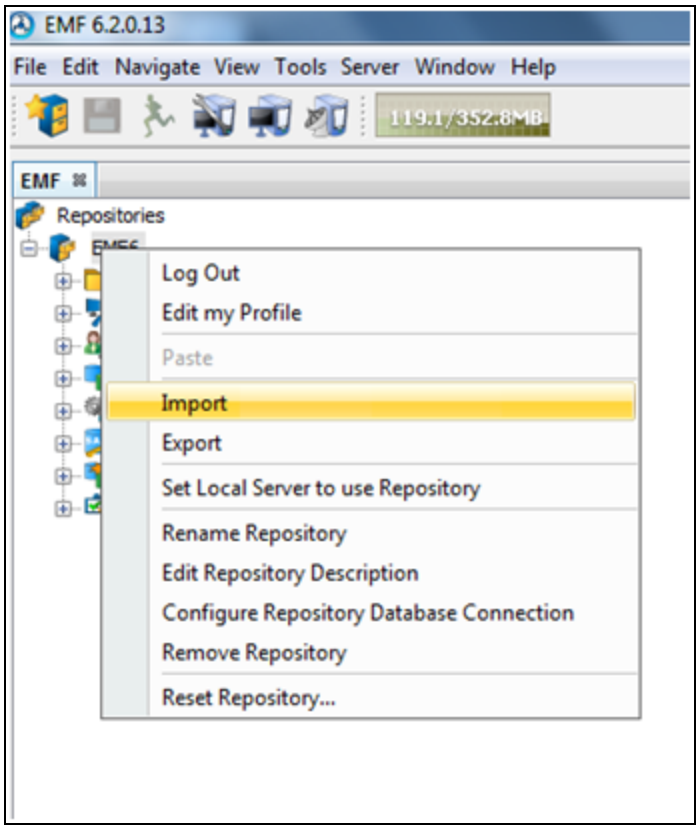
The prerequisite to Onyx-EMF integration is, install EMF version 6.2.0.13 and create an EMF repository. Refer to the EMF documentation for installation and how to create new repository.

To configure Onyx-EMF integration:

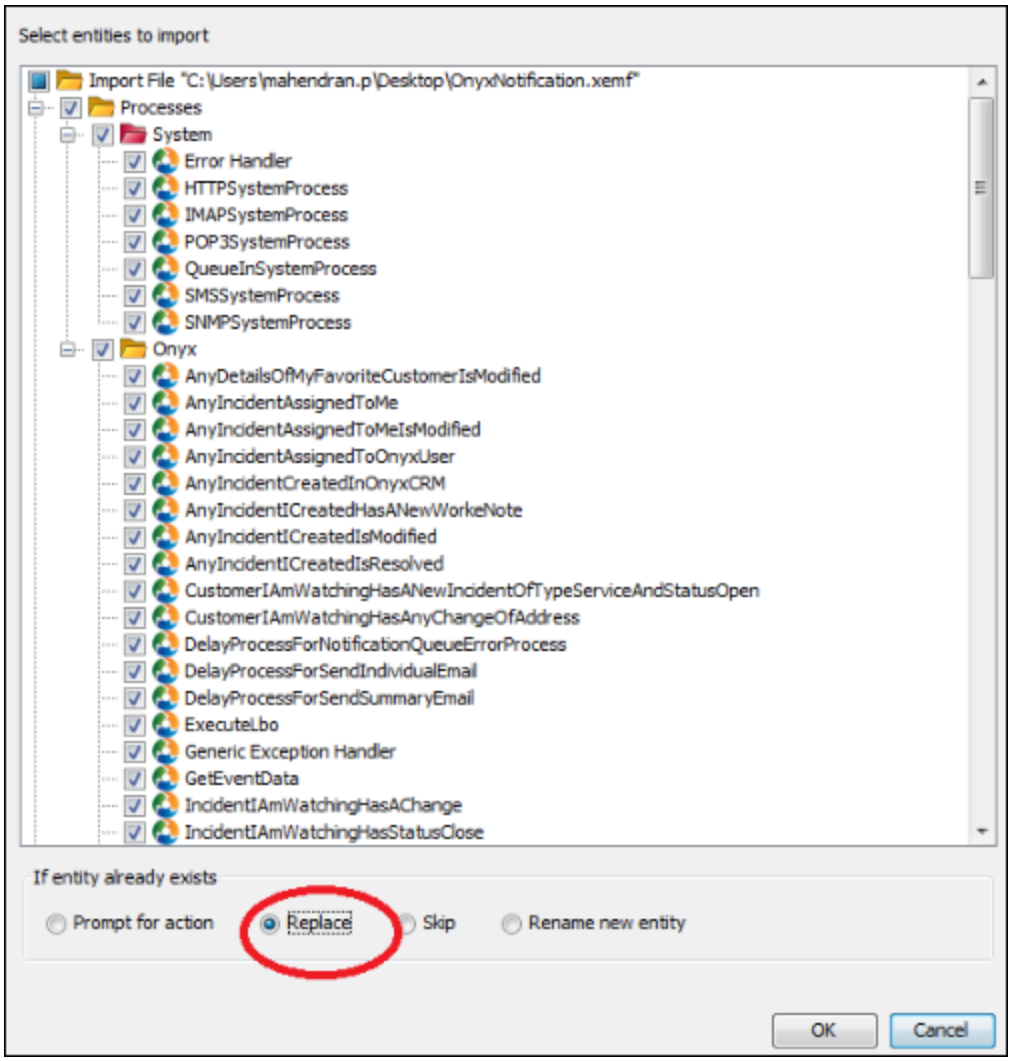
1. Login in to the repository.

> **Note:** To configure EMF, in the **Audit Log Configuration** dialog uncheck the option **Full logging of all changes** before the import of EMF processes.

2. Right-click on the repository name and import the file `Platform Components\EMF Server\EMFProcess\OnyxNotification.xemf`



3. If there is a prompt to replace certain entities, select Replace and click on **OK**.



Onyx-EMF Configuration

To install EMF the following processes need to be configured:

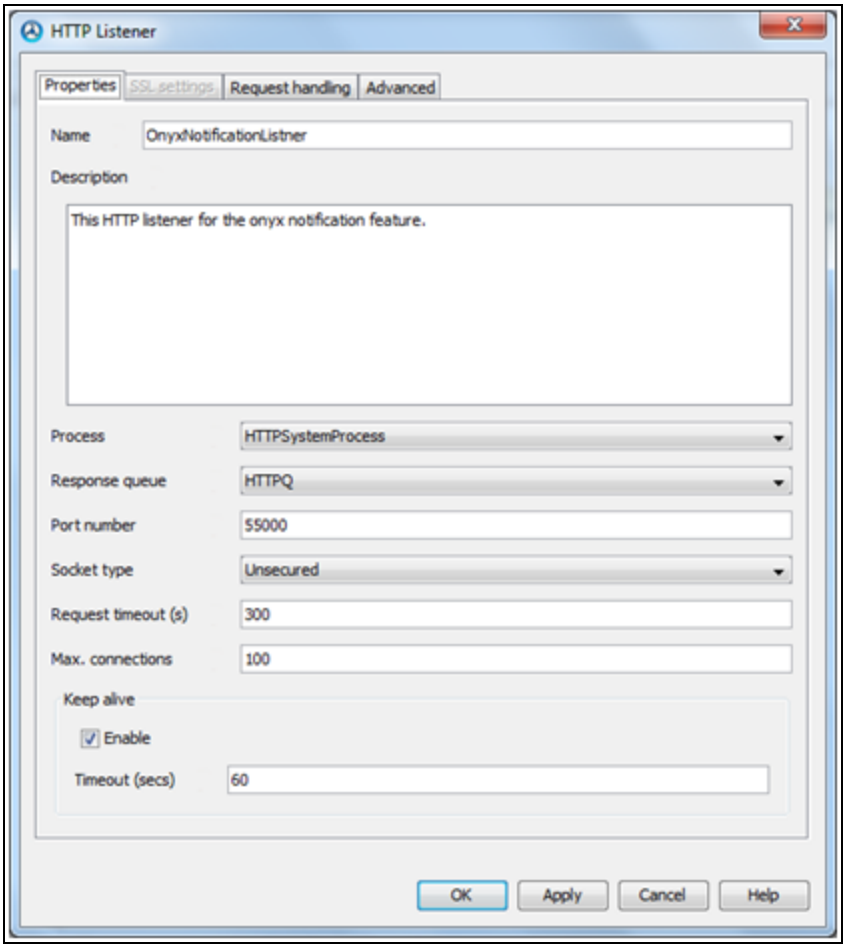
- OnyxNotificationListener
- OnyxEmfDBConnection
- OnyxExchangeServer

OnyxNotificationListener

The **OnyxNotificationListener** (HTTP Listener) in EMF server listens for incoming requests from ONS for any notification events.

To edit the properties of OnyxNotificatioListner:

1. Right-click on the process **OnyxNotificatioListener** and click on **Edit**.
2. In the **HTTP Listener** dialog, the out-of-the box port number for ONS to EMF communication is 5500. Ensure the same port number is updated in the **BrokerUrl** column in the **NotificationType** table.
3. Click **OK**.

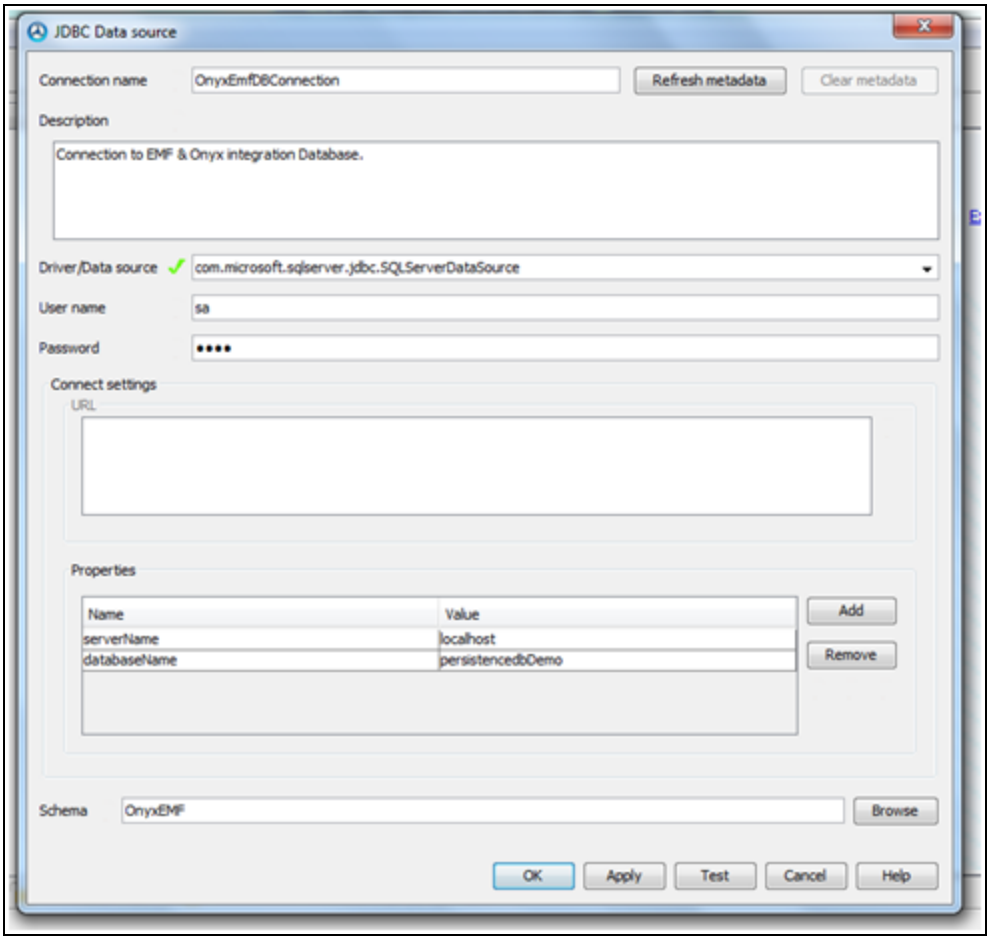


OnyxEmfDBConnection

JDBC connections are used to connect to Persistence DB data sources. Prior to setting up an EMF process, you must configure connections to access the PersistenceDB data sources.

To edit the properties of OnyxEmfDBConnection:

1. Right-click on the process **OnyxEmfDBConnection** and click on **Edit**.
2. In the **JDBC Data source** dialog, out-of-box database name is **persistencedbDemo**. In the Properties section enter the **serverName** and **dabaseName**.
3. Click **OK**.



OnyxExchangeServer

Onyx CRM uses SMTP Services to send notification emails.

To edit the properties of the OnyxExchangeServer.

1. Right-click on the process **OnyxExchangeServer** and click on **Edit**.
2. In the **SMTP Service** dialog, edit the required fields and click **OK**.

EMF Notifications

The notification types that can be edited in Onyx:

Individual Email template

A single email is sent for each event that occurs. An Individual email template is a html template for each notification type.

To edit an Individual email template for the notification type **Any incident assigned to me**:

1. Expand the **Onyx** folder.
2. Double click to open the **AnyIncidentAssignedToMe** process.
3. Find and open the module **MSG_EMAIL_BODY** by double clicking on the module name.
4. For example if you wish to change the ID field to Description, open the module **MSG_PARAM_LOOKUP_XML** and add the field which needs to convert ID to Description and use the same in the module **MSG_EMAIL_BODY**.

Summary Email template

A single summary email is sent for all events that occur in the day. Summary email body templates are based on LBO names. Out-of-box templates are Incident, Individual, Company, Task and Worknote.

Corresponding module names are **MSG_EMAIL_BODY_INCIDENT, MSG_EMAIL_BODY_INDIVIDUAL, MSG_EMAIL_BODY_COMPANY, MSG_EMAIL_BODY_TASK, and MSG_EMAIL_BODY_INCIDENT_WORKNOTE**.

To update an incident LBO notification type body template:

1. Expand the Onyx folder.
2. Double click to open the SaveSummaryEmailTemplate process.
3. Find and open the module MSG_EMAIL_BODY_INCIDENT by double clicking on the module name. Add the custom fields for EMF body.
4. Update a subject in the header column of on_summary_email_subject table in PersistenceDB

Summary emails - Schedulers

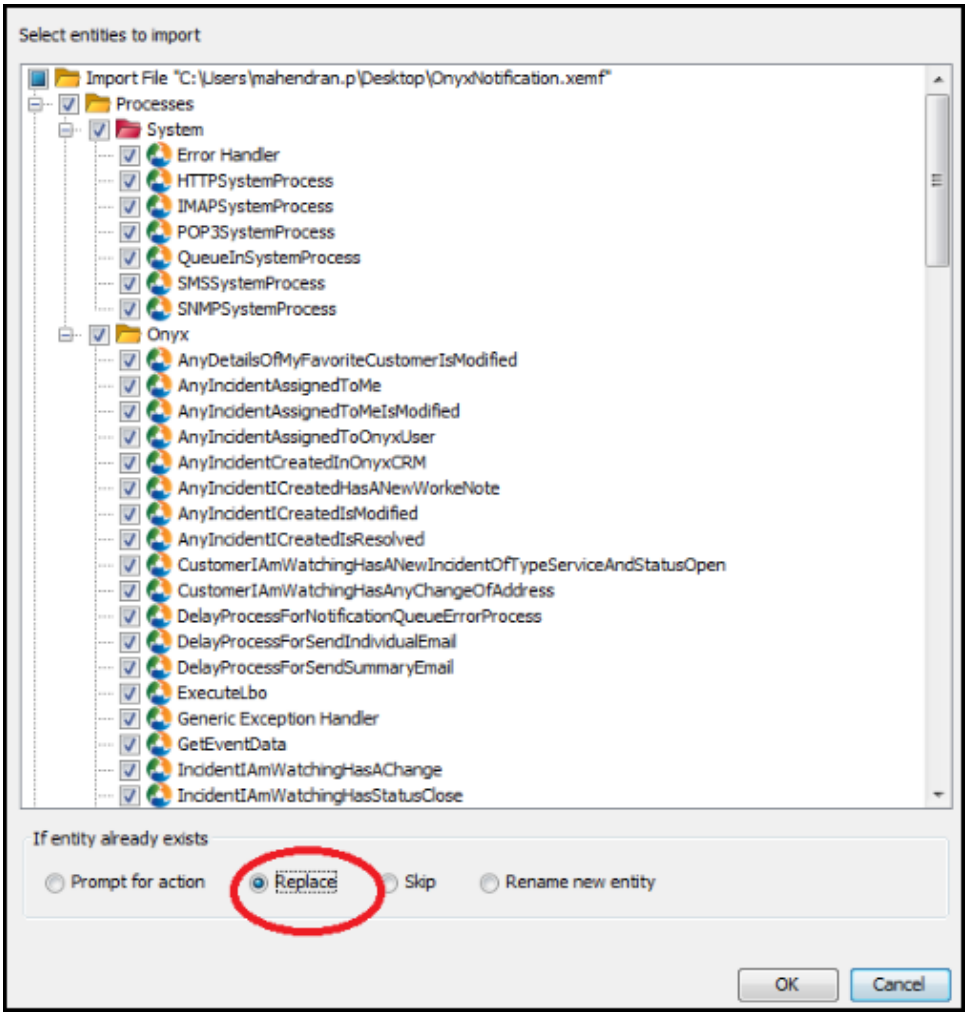
Scheduling Summary email contains the following notifications:

- SendPrivateSummaryEmail for private notifications
- SendPublicSummaryEmail for public notifications

To schedule a PrivateSummaryEmail to process and send emails:

Note: Once schedule time is set for summary email, restart the EMF server to see the changes immediately. If you do not restart, the changes are reflected 5 minutes after the scheduled time lapse.

1. Expand the **Onyx** folder.
2. Open **SendPrivateSummaryEmail** by double clicking on the process name.
3. Find and open the module **Schedule** by double clicking on the module name.
4. Update Start and End fields in the Repeating section as per the requirement.
5. Update start time **MSG_START_TIME** module with the value of **Start** field which was configured in previous step.
6. The time frame to run and send the Summary email can be defined in the module **MSG_END_TIME**. In this module the time has to be given in the minutes. Out-of-box this module runs for 4 hours.



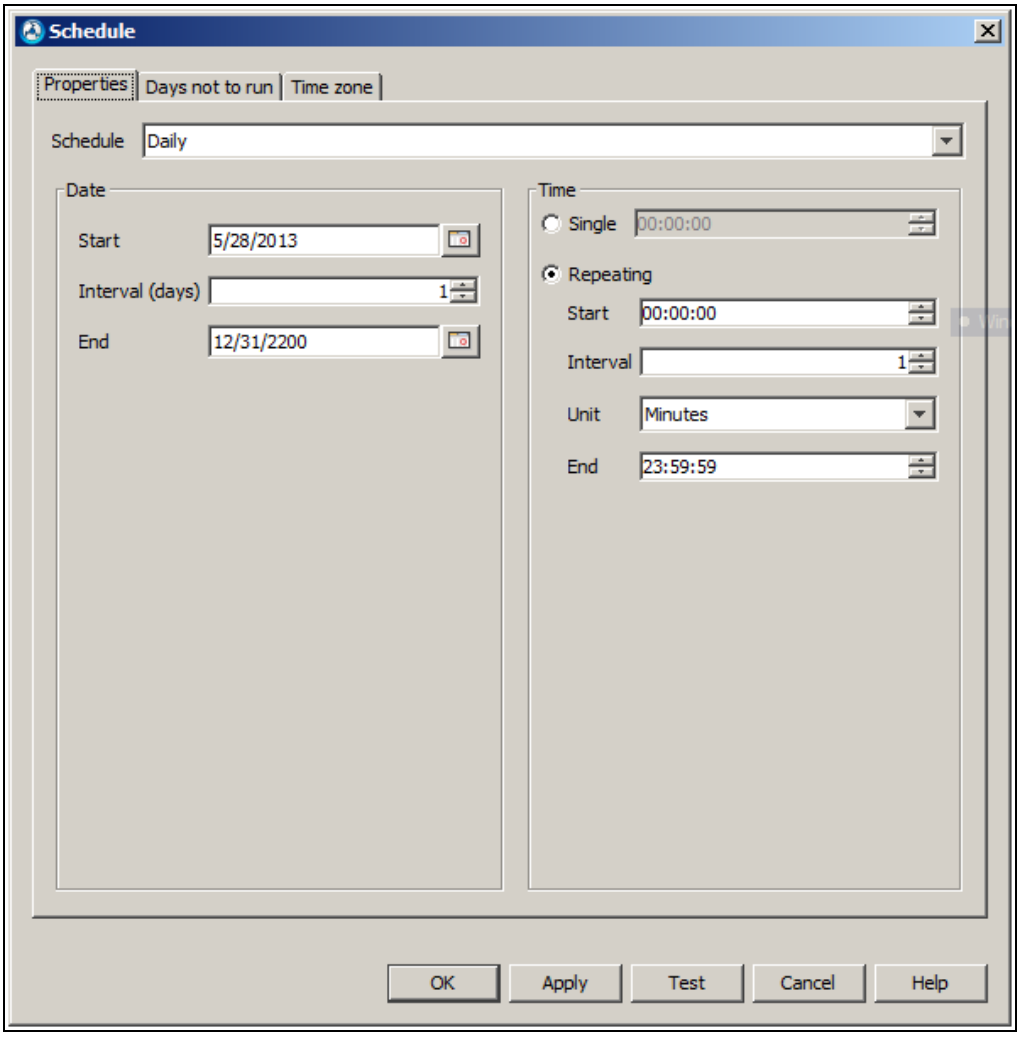
To schedule a PublicSummaryEmail to process and send emails:

1. Expand the **Onyx** folder.
2. Open **SendPublicSummaryEmail** by double clicking on the process name.
3. Find and open the module **Schedule** by double clicking on the module name.
4. Update Start and End fields in the Repeating section as per the requirement.

5. Update start time **MSG_START_TIME** module with the value of **Start** field which was configured in previous step.
6. The time frame to run and send the Summary email can be defined in the module **MSG_END_TIME**. In this module the time has to be given in the minutes.

To schedule a SendIndividualEmail to process and send emails:

1. Expand the **Onyx** folder.
2. Open **SendIndividualEmail** by double clicking on the process name.
3. Find and open the module **Schedule** by double clicking on the module name.
4. Update Start and End fields in the Repeating section as per the requirement.



EMF Configuration Notification Queue Cleanup

The EMF process **NotificationQueueCleanUp** will move records from the queue table to the completed queue table. This will ensure that the master table always has records to be processed, thereby improving performance when the table is queried as well. Technically the record from the tables **on_notification_queue** and **on_notification_queue_extension** are moved to **on_notification_queue_complete**.

Note: It is recommended to schedule all EMF Notification queue clean up processes during non productive hours.

PrivateSummaryQueueCleanUp

The EMF process **PrivateSummaryQueueCleanUp** will move records from the queue table to the completed queue table. This will ensure that the master table always has records to be processed, thereby improving performance when the table is queried as well. Technically the record from the tables **on_pri_summary_email_bdy_que**, **on_pri_summary_email_bdy** and **on_pri_summary_email_que** are moved to **on_pri_summary_email_bdy_que_cmp**, **on_pri_summary_email_bdy_cmp** and **on_pri_summary_email_que_cmp** respectively.

PublicSummaryQueueCleanUp

The EMF process **PublicSummaryQueueCleanUp** will move records from the queue table to the completed queue table. This will ensure that the master table always has records to be processed, thereby improving performance when the table is queried as well. Technically the record from the tables **on_pub_summary_email_body** and **on_pub_summary_email_queue** are moved to **on_pub_summary_email_bdy_cmp** and **on_pub_summary_email_que_cmp** respectively.

IndividualQueueCleanUp

The EMF process **IndividualQueueCleanUp** will move records from the queue table to the completed queue table. This will ensure that the master table always has records to be processed, thereby improving performance when the table is queried as well. Technically the record from the tables **on_individual_email_queue** and **on_individual_email** are moved to **on_individual_email_queue_cmp** and **on_individual_email_complete** respectively.

NotificationQueueErrorProcess

The EMF process **NotificationQueueErrorProcess** will re-execute the notification queue if there is an error record of three times in a given interval.

OGS URL in EMF

The table on_system_parameter has the system parameters required for the notification feature. Once the EMF installation is complete the following column needs to be updated.

Column Name	Description
site_id	Numeric number of the site id
language_ code	Language code
ogs_url	URL of OGS required to communicate from EMF to OGS. It is used to get email template data and execute business rules
oep_url	URL of OEP used in the email template hyperlinks that open the corresponding record
application_ name	Onyx application name
user_id	User id EMFSa to connect to OEAS. The user id can be a windows NT user
password	Password to connect to OEAS (Should be encrypted using PasswordEncrypt tool)
update_date	The date the record was last updated

EMF Exception Handling and Logging

Any exception created in the Onyx EMF Processes will be captured by a single exception handler. The Exception Handler can do the following functions:

- Log the exception details to file

  The log files by the exception handler are created in the **C:\Program Files (x86)\Aptean\EMF\Server\outputs\OnyxEMFLogs** folder. The log files contain details of the exception and the name of the process in which the exceptions are generated.

To change the path of the log file:

1. Open the **Onyx\LogHelper** Process.
2. Modify the text formatter, **Log Files Folder Path** to change the path of the output folder.

- Send an email to the administrator

Any exceptions that occur in the Onyx EMF integration processes are sent to the administrator. The recipient for the exception emails are defined in **Recipients Administration\Recipients\ Onyx EMF Integration Administrator**. To change the email address, the recipients email address should be changed.

Whether an email should be sent or not can be toggled by modifying the LogHelper process.

1. Open the **Onyx\LogHelper** Process.
2. Modify the text formatter, **Send Exception Details as Email to 0**.

Associating process to generic exception handler

To set the generic exception handler to any process:

1. Open the **EMF** Process.
2. Right-click to select the option **Edit process properties**.
3. Select the **Error Handler** tab. Set the Enable error handling to **True**.
4. Select the option **Generic Exception Handler** in the Onyx folder.

Enable user message logging

Processes in EMF can be associated to log user messages to file. The user messages contain information regarding the execution steps and other key data, which can be used to debug in case of an error. All EMF Onyx Integration processes are enabled with logging and user messages while executing a process.

To write user messages to file:

1. Open the **LogHelper** Process.
2. In the Process, open the module **Log User Message to File**.
3. Set the value to 1.

To write user messages to EMF Logs:

1. Open the **LogHelper** Process.
2. Right-click and select the option **Edit process properties**. Set the Log user messages to **True**.

Use the logging in case of exception

During the execution of a process if an exception occurs an email will be sent to the administrator. Following steps should be followed to figure out the actual problem:

1. Enable user message logging (File and EMF Logs).
2. Get the Statistics logging from EMF which is enabled by default on all processes.
3. The two files will enable us to identify which event, process and module the exception was raised.

5

Navigator

Overview

Use this information to configure and customize Navigator and Onyx Mobile to suit your organization's requirements. For information on modifying OGS configurations, see Modifying the OGS Configuration File

The tasks explained in this topic are:

- Administering Navigator
- Modifying Navigator Action Menu
- Customizing Navigator

Administering Navigator

You can determine the fields that users can search by and view in the result grid. You can change field captions to suit your business needs. You can also change the properties for fields that are displayed in the result grid.

The tasks explained in this topic are:

- Navigator Search Criteria Administration
- Navigator Search Result Grid Administration

Navigator Search Criteria Administration

For each Search Type available in Navigator, you can limit the fields that will be available to users to perform a search. You can also change the default field names to ones that are better suited to your organization's requirements. These options are available only if you are logged on as a user with administrative privileges.

- Determine available search criteria
- Understanding hierarchical relationships
- Edit a field name
- Customizing the Navigator Static Data

Determine available search criteria

To determine search criteria

1. Open the **Search Criteria Field Selection and Administration** window. To do this, in the Navigator **Search Criteria** window, select a **Search Type**, click  , and then click Search Criteria.

The **Selected** column lists the fields that are currently available to users as default **search criteria**. These fields display in the **Search Criteria** window when you select the search type. The **Available** column lists the fields that can be included as search criteria. The **Disabled** column lists the fields that cannot be included as search criteria.

2. To make a field available as a search criterion, select and drag it to the **Available** column.
3. To include a field as a default search criterion that displays in the Search window, select and drag it to the **Selected** column.
4. To remove a field as a default search criterion, but to keep it available for users to add, select and drag it to the **Available** column.
5. To completely remove a field as a search criterion, select and drag it to the **Disabled** column.

   Some fields may share hierarchical relationships with other fields. To understand how such fields are moved from one list to another, see Understanding hierarchical relationships.

6. Save your changes or click  to restore the default search criteria, and close the window.

Understanding hierarchical relationships

The search criteria may include fields that are hierarchical, that is, the values in those fields depend upon values entered in other fields. For example, since the values for the field State depend on the value for the field Country, Country is the parent field for the dependent field State. This hierarchy can exist at several levels; therefore, a field can be a parent field as well as a dependent field. For example, in a hierarchy of Resolution Code 1 - Resolution Code 2 - Resolution Code 3, the field Resolution Code 2 is a parent for the field Resolution Code 3, but a dependent for the field Resolution Code 1.

You can move a parent field from one column to anyone of the other two columns. When you move a parent field, the associated child fields move along with the parent field, but you cannot move a child field alone.

Edit a field name

To edit a field name

1. Open the **Edit Field Properties** window. To do this, in the Navigator **Search Criteria Selection and Administration** window, select a field and double click .
2. In the **Caption** box, type the desired name for the selected field.
3. To make the field Required, select the Required check box in the Edit Field Properties window.
4. Save your changes and close the window. The **Search Criteria Selection and Administration** window lists the field with the changed name.

Customizing the Navigator Static Data

You can customize the Navigator static data for drop down fields on the **Search Criteria** window and associated results on the **Search Result** window based on your organization's preferences.

To customize the Navigator static data

1. Navigate to **Program Files** > **Onyx** > **EmployeePortal** > **QuickSearch** > **resources** > **custom**, and open the **Navigator_Static.json** file in edit mode.
2. Add the values of `Displayvalue` and `DisplayName` attributes for the cache names based on your organization's preferences. For more details about cache names that can be customized through the **Navigator_Static.json** file, click List of Cache Names.
3. Save and close the **Navigator_Static.json** file.
4. Log on to **OEP** and verify the changes.

List of Cache Names

Following table shows the details of cache names that can be customized through the **Navigator_Static.json** file:

Cache Name	Use Type	Search Criteria Window	Search Result Window
gender	0 and 2	✓	✓
email.followup	0	✓	
email.draft	0	✓	
combodefault	0	✓	
rld.data	0	✓	
rld.period	0	✓	
attachment.owner	0 and 2	✓	✓
script.status	0 and 2	✓	✓

Note: If you do your own customization for Search Result window, the customization done through the **Navigator_Static.json** file does not pass.

Navigator Search Result Grid Administration

For each Search Type available in Navigator, you can limit the fields that users can view in the result grid. You can change the default field names to ones that are better suited to your organization's requirements, and make fields available for inline editing. These options are available only if you are logged on as a user with administrative privileges.

- Determine the fields to display in search results
- Edit result grid field properties
- Customizing the tile view of search results for an entity
- Modify system limit for records displayed in Navigator
- Modifying the action menu display

Determine the fields to display in search results

To determine the fields to display in search results

1. Open the **Search Results Field Selection and Administration** window. To do this, in the Navigator **Search Criteria** window, select a **Search Type**, click  and then click Result Columns.
2. The **Selected** column lists the fields that are currently displayed in the **Search Results** window. The **Available** column lists the fields that can be included in the **Search Results** window, while the **Disabled** column lists the fields that are currently not available for selection.
3. fTo include a field as a selected field, select and drag it to the **Selected** column.
4. To make a field available for users to choose, select and drag it to the **Available** column.
5. To completely remove a field so that it is not available for users to choose, select and drag it to the **Disabled** column.
6. In the **Options** column, select the fields to sort the search results by. The **Sort by** and **Then by** drop-down lists include all the fields from the **Selected** column. You can select up to three fields to sort results by.
7. In the **Maximum Rows** field, enter the maximum number of rows that a search for the selected search type should return from the database. For example, if you enter a value of 100, the search returns the first 100 records that match the criteria you specified.
8. The default value in this field is 300. The maximum value that you can enter depends on the limit set by the system administrator in the OnyxWindowsService.exe.config file. For information on modifying this limit, see Modifying maximum records displayed in Navigator.
9. In the Page Size, enter the maximum number of records to be displayed per page.
10. Save your changes or click  to restore the default search criteria, and close the window.

Edit a field name

To edit a field name

1. Open the **Edit Field Properties** window. To do this, in the Navigator Search Results Field Selection and Administration window, select a field and double click.
2. In the Caption box, type the desired name for the selected field.
3. Save your changes and close the window. The Search Results Field Selection and Administration window lists the field with the changed name.

Customizing the tile view for search results of an entity

You can customize the tile view for search results of an entity based on your organization's preferences.

To customize the tile view

1. Navigate to **Program Files** > **Onyx** > **EmployeePortal** > **QuickSearch** > **resources** > **custom**, and open the **EntityTileTemplate.json** file in edit mode.
2. Add `html` attributes for the entities based on your organization's preferences.

Note: You need to add all custom entities and its html attributes in the **EntityTileTemplate.json** file to get the successful search results.

Example:

Following example shows the customization of tile view for search results of a custom entity .

```
"Entity": "Customers",
"HTML": "<div class=\"MainTile\"><div><div class=\"Company-
First-Field\">{customerName:htmlEncode}</div><div
class=\"Company-icon\"></div></div><div class=\"Company-Field\">
{customerId:htmlEncode}</div><div class=\"Company-Field\">
{url:htmlEncode}</div><div class=\"Company-Field\">
{stateDesc:htmlEncode}</div><div class=\"Company-Field\">
{countryDesc:htmlEncode}</div></div>"
```

3. Save and Close the **EntityTileTemplate.json** file.
4. Log on to **OEP** and verify the changes.

Modifying Navigator Action Menu

You can configure the Navigator Action menu to display in a horizontal or vertical format based on your organization's preferences. The default setting for this is horizontal format.

To configure Navigator Action Menu display:

1. In OES System Parameter Administration, select the parameter code GAMType.
2. In the Parameter Value field, enter text to determine how the action menu should be displayed.
   - Enter the text Menu to display the action menu in a vertical format.

Note: Be sure to enter the exact text Buttons or Menu in the Parameter Code field, as per your preference. If you enter any other value in this field, the action menu is displayed in horizontal format.

3. Save the value you entered.

Customizing Navigator

You can customize Navigator to incorporate the modifications you made to your Onyx system and to enable users to efficiently search for and edit records.

The tasks explained in this topic are:

Prerequisites

Before you begin customizing Navigator:

- Install and configure Onyx, along with all the optional components that you need.
- Modify the system as desired.
- In the Onyx database, create the objects and table columns that you want to search by in Navigator.

For information on installing and configuring Onyx, see the Onyx Installation Guide.

Adding a sample custom entity

This release contains sample scripts that you can use to add Customers as a custom entity in Navigator. You can also use these scripts as a template to create your own scripts.

The following scripts are available in the Customer Entity folder, available under Customization Support>Database Server>Navigator Custom Entity Scripts within your installation package:

- Customer_navigator_view.sql: Creates the entity view for Customer in the Onyx database.
- Customer_navigator_entity_master.sql: Updates the navigator_entity_master table in the Persistence database.

- Customer_navigator_entity_master_ml.sql: Updates the navigator_entity_master_ml table in the Persistence database.
- Customer_navigator_entity_field_master_and_ml.sql: Updates the navigator_entity_field_master and navigator_entity_field_master_ml tables in the Persistence database.
- CustomerUIResources.sql: Creates a new UI resource.

To add Customer as a custom entity:

1. Run the above scripts in the order listed above. This creates the required views in the Onyx database, inserts default data in the Persistence database, and creates UI resources.
2. Modify the Onyx Gateway Service configuration.
3. Modify Result Grid properties.
4. Add a Navigator menu or bookmark item.
5. Enter resource string values.
6. Add custom fields if required.
7. Add a custom action button if required.
8. Enable inline editing if required.

About Navigator Search Operators

You can change the default search operator for all Navigator Search Criteria. Use this information to understand the operators that are currently available, and how to set a specific search operator as the default operator.

The tasks explained in this topic are:

- Navigator Search Operators
- Setting the default search operator

Navigator Search Operators

The following is the list of Navigator search operators and the corresponding codes and values from the navigator_operator table in Persistence database.

Operator_Code	Operator_Description	Operator_value
LKE	Begins With (%)	%
ISN	Is Empty (ET)	ISN
ISNN	Is Not Empty (NE)	ISNN
EQ	Equals To (=)	=
NEQ	Not Equal To (<>)	<>
GT	Greater Than (>)	>
LT	Less Than (<)	<

Operator_Code	Operator_Description	Operator_value
GE	Greater or Equals To (>=)	>=
LE	Lesser or Equals To (<=)	<=
BTWN	Is Between (-)	-
IN	IN	IN
CNT	Containts	%
NIN	NOT IN	NOT IN
TOD	Today	Today
TWK	ThisWeek	ThisWeek
TMNTH	ThisMonth	ThisMonth
TQTR	ThisQuarter	ThisQuarter
TYR	ThisYear	ThisYear
TMRW	Tomorrow	Tomorrow
NWK	NextWeek	NextWeek
NMNTH	NextMonth	NextMonth
NQTR	NextQuarter	NextQuarter
NYR	NextYear	NextYear
LWK	LastWeek	LastWeek
LMNTH	LastMonth	LastMonth
LQTR	LastQuarter	LastQuarter
LYR	LastYear	LastYear
ANYT	AnyTime	AnyTime
YTD	Yesterday	Yesterday
internalContactIds	Internal Contact Inner Join	internalContactIds
tags	Keyords Inner Join	tags
INORIN	INORIN	INORIN

Setting Default Operator in Navigator

To set the default operator in Navigator, you must modify the value in the default_selected column of the navigator_operator table in Persistence database.

To set the default operator in Navigator

1. In Persistence database, run the following SQL query:

select * from navigator_operator

This will display the list of operator codes along with their description. The current default operator has the value '1' in the default_selected column. Note the operator codes for the current default operator as well as the new operator that you want to set as default.

2. Run the following query to change the default operator as desired.

   - Replace [new_operator_code] with the code corresponding to the desired default operator

     update navigator_operator set default_selected=1 where operator_code='[new_operator_code]'

   - Replace [old_operator_code]with the code corresponding to the current default operator in order to remove it as the default

     update navigator_operator set default_selected=0 where operator_code='[old_operator_code]'

3. Log on to OEP and verify that the newly set operator is displayed as the default operator in Navigator Search Criteria.

Adding Custom Search Type

You can add a new search type (custom entity) in Navigator that enables you to run a search for an object you created in Onyx.

The tasks explained in this topic are:

- Modifying Onyx Gateway Service Configuration
- Inserting Custom Entity Information in Persistence Database
- Inserting Custom Field Information in Persistence Database
- Inactivating an Entity in Persistence Database
- Creating Entity View in Onyx Database
- Modifying Result Grid Properties
- Adding Navigator Menu or Bookmark Item
- Entering Resource String Values

If a custom entity is no longer relevant to your business, you can remove it from Navigator.

Modifying Onyx Gateway Service Configuration

To modify the Onyx Gateway Service configuration:

1. On the **OEAS** server, stop the **Onyx Gateway Service**.
2. Back up the **Onyx Gateway Service installation** folder and all its contents. The default installation path is C:\Program Files\Onyx\AppServer\Applications\Onyx.
3. In the OGS installation folder, double-click **CustomEntityHelper.exe**.

The **Custom Entity Helper** window appears.

4. In the **Entity Name** box, type a name for the custom entity that you want to create. Ensure that the value you enter here matches the value you enter in the **entity_name** field of the **navigator_entity_master** table.
5. Click the **Create Entity** button. This creates a file in the OGS folder with the name <EntityName>.dll, where <EntityName> is the name you entered.
6. After you have created all the entities that you want to use, recreate the CMService.dll to include the new entities.
   i. In the **All Custom Entity Names** box, enter all the existing custom entity names, separated by commas.

Important: Recreating the CMService.dll removes all previously created custom entities, so you must enter all the custom entity names created so far each time you recreate the CMService.dll.

---

   ii. Verify that one <EntityName>.dll file exists for each custom entity name that you entered in the box.
   iii. Click the **Create CMService** button to recreate the CMService.dll including all the custom entities.
7. From the Onyx Gateway Service folder, open the Onyx Gateway Service configuration file, and search for the comment.'<!—insert custom entity endpoints here -->'.
8. Insert the following endpoint node under the comment '<!—insert custom entity endpoints here -->', where <EntityName> is the value you entered for the entity in step 4.

```
<endpoint address="ServiceGateway/<EntityName>/"binding="webHttpBinding"
contract="CmService.Service.I<EntityName>"/>
```

9. Start **Onyx Gateway Service**.
10. Verify the configuration to test that the custom entity is being called. To do this, in **Microsoft Internet Explorer**, type the following URL, replacing <entity-name> with a custom entity name you created.

```
http://<app-server-name>:69/ServiceGateway/<entity-name>/Search/?
```

Onyx should return an error message in the following format, indicating that the call was successful, but that the search failed due to a lack of additional parameters.

```
<returnXml>
<methodStatus>
<statusType>failure</statusType>
<error>
<title>Navigator</title>
```

```
<caption>Custom Entity search failed</caption>
<messageHandle>00000000-0000-0000-0000-000000000000</messageHandle>
<severity>failure</severity>
<dialog moreButton="False" />
</error>
</methodStatus>
</returnXml>
```

Inserting Custom Entity Information in Persistence Database

After creating the view for the custom entity, update the corresponding tables in the Persistence database. Read a description and see example values for each table that needs to be modified.

To insert custom entity information in the Persistence database:

1. In the **Persistence** database, right-click the **navigator_entity_master** table and select **Edit Top 200 Rows**.
2. Scroll to the bottom of the displayed result grid, and insert one row for the entity that you created a view for.
3. Select the **navigator_entity_master_ml** table and repeat steps 1 and 2.

Inserting Custom Field Information in Persistence Database

Update the Persistence database with information for each field of each entity that you want to view in the Search Criteria and Search Results windows. Additionally, you must also enter information for each field that is marked as Required for the Logical Business Object corresponding to the entity. Read a description and see example values for each table that needs to be modified.

To insert custom field information in the Persistence database:

1. In the **Persistence** database, right-click the **navigator_entity_field_master** table and select **Edit Top 200 Rows**.
2. Scroll to the bottom of the displayed result grid.
3. Insert one row for each field that you want to add as a Search Criteria in Navigator.
4. Insert one row for each field that you want to add as a Result Grid column in Navigator.

> **Note:** For a field to appear both as a search criterion and as a result grid column, insert two distinct rows for the field.

5. Insert one row for each field that is marked as Required for the Logical Business Object (LBO). For these fields, enter the value TRUE in the is_transaction column, the value 2 in the use_type column, and the value 0 in the field_state column of the navigator_entity_field_master table.

Note: Do not add a separate row if you've already added a row to include the field as a Result Grid field. However, ensure that you enter the value TRUE in the is_transaction column for such a field.

6. Insert two rows for the field primaryId with the value TRUE in the is_transaction column and the value 0 in the field_state column. In the use_type column, enter the value 0 in one row, and the value 2 in the second row.
7. Right-click the **navigator_entity_field_master_ml** table, and repeat steps 2 to 7 above.

Important: Every field that you add to these tables must also exist in the corresponding view.

Inactivating an Entity in Persistence Database

You can inactivate an entity if it is no longer required. An inactive entity or field is not available in Navigator to perform a search.

To inactivate an entity:

- In the **navigator_entity_master** table in the **Persistence** database, set the value for the **IsActive** field for the search type to 0.

Creating Entity View in Onyx Database

Create the entity view in the Onyx database to include the custom object and fields. For a list of the default views available in the Onyx database, see to Entity Views.

To create an entity view:

1. In the **Onyx database**, identify the object for which a view needs to be created. You can select an existing object or create a new object.
2. Create a view for the object.

   Include in the view, all the fields that you want to use in the Search Criteria and the Result Grid areas of Navigator. Also include all fields that are required fields for the Logical Business Object (LBO). Ensure that you include languageCode as a field in each view that you create.

3. If you are adding a reference field to the view, you must include one more line in the view, in the following format:

   [ReferenceTableAlias].[field_name] AS [FieldAlias]Sort

   This will enable the selected Sort By option to be saved when the query is saved.

4. Execute the newly created view against the Onyx database.
5. Be sure that the newly created view has **Select Grant** permission for the **OnyxLMViews** role.

Modifying Result Grid Properties

You must modify the Navigator Result Grid properties for a custom entity to ensure that the corresponding page opens when you click a column in the Navigator Result Grid.

To understand the process that occurs when a user clicks a column in the Navigator Result Grid, see Opening a page from the Navigator Result Grid.

To modify result grid properties:

1. Create a new switch case in the JavaScript function for the Custom Entity with two actions, one for a hyperlinked column, and one for a non-hyperlinked column.
2. Get all the required parameters to be passed.
3. Call the function to open the CM page of custom entity/object and pass the required parameters, if any.

Example of a switch case for an entity 'Customers'

```
case "CUSTOMERS":
//Check whether the request is from a hyperLink field or normal field
if (bHyperLink != "") {
var sCustomerId;
var sCustomerType;
if (bHyperLink == "true") {
if (oRootNode.selectSingleNode("customerId") != null)
sCustomerId = oRootNode.selectSingleNode("customerId").text;
if (oRootNode.selectSingleNode("customerType") != null)
sCustomerType = oRootNode.selectSingleNode("customerType").text;
if (sCustomerId != "")
jsLoadObjectCustomer(sCustomerId, sCustomerType, "");}
else if (bHyperLink == "false") {
if (oRootNode.selectSingleNode("customerType") != null)
sCustomerType = oRootNode.selectSingleNode("customerType").text;
if (sPrimaryId != "")
jsLoadObjectCustomer(sPrimaryId, sCustomerType, "");
}
}
break;
```

Opening a page from the Navigator Result Grid

The page that opens when you click a column in the Navigator Result Grid depends on whether the column is hyperlinked or not.

- If the column is hyperlinked, then the hyperlinked page opens.
- If the column is not hyperlinked, then the corresponding entity page opens.

For example, for the Product entity, a column Customer in the result grid is a hyperlinked column. If a user clicks the Customer column, then the Customer page should open. However, if a user clicks any other column in the Result Grid, then the Product page should open.

The following processes occur when a user clicks a column in the Result Grid:

1. XML is generated with the root element and all the properties of the selected entity.
    - When a user clicks a hyperlinked column, the hyperlinked entity is the root element in the XML.

    For example, Customer

```
<Customer> - root element
<Element1>value</Element1>
<Element2>value</Element2>
<Element3>value</Element3>
<Element4>value</Element4>
<Element5>value</Element5>
</Customer>
```

    - When a user clicks a non-hyperlinked column, the selected entity is the root element in the XML.

    For example, Product

```
<Product>- root element
<Element1>value</Element1>
<Element2>value</Element2>
<Element3>value</Element3>
<Element4>value</Element4>
<Element5>value</Element5>
</Product>
```

This XML is passed as a parameter to the function 'jsOpenEntity (entity)'. The default path for this file is C:\Program Files\Onyx\Employeeportal\powerpage\application_main.js

The function performs the following actions:

- Loads the XML
- Gets the Entity Object
- Gets the Entity Name

- Gets the common Parameter (generally the 'primaryId' of the object)
- Checks the value of the 'HyperLink' attribute
- Uses the switch case logic for each entity, to make a call to the appropriate function to open the corresponding page

Note: In the **Navigator Results List Administration** window, if you change the order and position of fields, your sorting preferences may be lost.

Adding Navigator Menu or Bookmark Item

You must modify the configuration file to add the custom entity as a menu or bookmark item to the Navigator menu on the Onyx header bar.

To add a custom entity as a menu or bookmark item:

1. On the OEP server, navigate to **C:\Program Files\Onyx\EmployeePortal**, and open the **Config.xml** file.
2. Navigate to the node - **<header:def id="HeaderBarRoot">**
3. Under that, navigate to the child node - **<header:item id="QuickSearch">**
4. Under this node, create a new entry to add the custom entity as a menu item.
5. Create another entry to add the custom entity as a bookmark item.

If you add more than 20 custom bookmarks and menu items to the Navigator menu on the Header bar, and the menu extends below the browser pane, some bookmarks or menu items may not be visible. To resolve this issue, remove the items that you do not require, ensuring that all items fit within the browser pane.

Example data for the Attribute Table for Menu and Bookmark items

Attribute Name	Description for Menu Items	Description for Bookmark Items
id	Enter the entity name followed by the keyword Search. For example, if the entity name is Customers, then the Id should be CustomersSearch	Enter the entity name followed by the keyword Bookmarks. For example, if the entity name is Customers, then the id should read as CustomersBookmarks. This value, which is case sensitive, maps the menu item of the entity to the bookmark(s).
ucf:uid	Globally Unique Identifier (GUID). Generate a new GUID and use the same. For information on generating GUIDs, refer to MSDN (http://msdn.microsoft.com/en-us/library/aa475087.aspx)	Globally Unique Identifier (GUID) – Generate a new GUID and use the same. Refer to the GUID Generation topic in MSDN to review the steps for generating GUIDs: http://msdn.microsoft.com/en-us/library/aa475087.aspx
popUp	Enter 0 (zero) as the value	Enter 0 (zero) as the value
flyout	Not applicable for menu items	Enter 1 (one) as the value

Attribute Name	Description for Menu Items	Description for Bookmark Items
url	Leave this blank	Leave this blank
target	Enter idFrameNavigator as the value	Enter idFrameNavigator as the value
targetArgument	Use the same value that you entered in the entitiy_name field in the navigator_ entity_field_master table in the Persistence DB. This value, which is case sensitive, maps the database to the CM application.	Leave this blank
captionId	To support localization for the caption of the menu item, add a new entry in the 'config.xml' file that is located under C:\Program Files\onyx\employeeportal\ in the OEP server: <string name="menu_ customerSearch" id="" text="Customer"/> Use the value of the 'name' attribute for captionId attribute. The value supplied as the 'text' attribute appears in the OEP.	To support localization for the caption of the bookmark item, add a new entry in the 'config.xml 'which is under C:\Program Files\onyx\employeeportal\eng on the OEP server: <string name="menu_ customerBookMarks" id="" text="Customer Bookmarks"/> Use the value of the 'name' attribute for captionId attribute. The value supplied as the 'text' attribute appears in the OEP.

Entering Resource String Values

Create one entry for each custom search type in the Navigator.Resources project to specify the label to identify the custom entity in tool tips and messages.

To enter resource string values:

1. In your development environment, ensure that you have installed the following programs on your computer:
   - The latest version of Microsoft Visual Studio 2012 with the latest updates available from Microsoft
   - Microsoft Silverlight 5.0
   - RadControls for Silverlight Q3 2013
2. In your Onyx 7.8 installation package, go to Customization Support\Web Server\.

   To modify resource strings for a specific language, use the installation package for the corresponding language.
3. From the **Web Server** folder, copy the **Navigator.Resources** project to the desired folder on your development computer.
4. From the **Navigator.Resources** project, open the solution file **Navigator.Resources.sln** in Visual Studio 2012.

5. In Solution Explorer, open the **LocalResource.resx** file.
6. Create one entry for each custom search type and enter values in the following format:
   - Resource String Name = <label_customentityname>, where the entity name is the value that you entered in the entity_name field of the Navigator_entity_master table.
   - Value = <desired display value in the installation language>

   For example, for the custom search type 'Customers', you can enter the following: Resource String Name = Label_Customer, and Value = Customers.
7. Set the configuration mode to Release. To do this, select **Build>ConfigurationManager option>Active Solution Configuration**
8. Build the solution.
9. In the Navigator.Resources project folder, navigate to **\Navigator.Resources\Bin\Release**, and copy the updated **Navigator.Resources.dll**.
10. Paste the Navigator.Resources.dll into the **ClientBin** folder within the OEP installation folder. The default location for the folder is C:\Program Files\onyx\EmployeePortal\QuickSearch\ClientBin.
11. Clear all cache and temporary Internet files on each client computer.

Important: Be sure to back up the modified project. You will need it when you want to make further modifications and when you want to upgrade to a newer version of Onyx.

Adding Custom Field

A custom field is a search criteria or result grid field that you add to a default or to a custom Navigator search type.

To add a custom field to a Navigator search type, you must:

- Insert Custom Field Information in Persistence Database
- Modify Entity View in Database

To make a field a hyperlink, you must:

- Insert Hyperlink Information in Persistence Database
- Modify Result Grid Properties

If a field is no longer relevant to your business, you can remove it from Navigator.

Modifying Entity View in Database

Modify the desired entity view to add a new field. You can modify a default view, or a custom view that you created for a new entity. For a list of the default views available, refer to Entity Views.

To modify the entity view:

1. In the Onyx database, identify the table columns to include in the view.

   Be sure to include those table columns in the view that are required to perform any operation for the entity in Onyx.

2. Expand **Views**.
3. Right-click the view that you want to modify, and select **Script View As > ALTER To > New Query Editor Window**.
4. Add the custom fields for the selected view at the end of the fields list in the SELECT statement, in the following format:

   [TableAlias].[field_name] AS [FieldAlias],

   For example, c.company_id AS primaryId, where the field_name company_id from the Company table is displayed as the primary ID.

5. If the custom field that you are adding is a reference field, then you must include one more line in the view, in the following format:

   [ReferenceTableAlias].[field_name] AS [FieldAlias]Sort

   This will enable the selected Sort By option to be saved when the query is saved.

6. Execute the modified view against the Onyx database.

Inserting Hyperlink Information in Persistence Database

When you create a custom entity, you can hyperlink certain fields in the Navigator Result Grid to open the corresponding Onyx page, or an external webpage. An entity can have several hyperlinked

fields. The **primary_linked_field_name** and **secondary_linked_field_name** columns hold values that determine which page is opened when you click a link.

To create a hyperlinked field:

1. In the **navigator_entity_field_master** table:
   a. Select the row corresponding to the field that you want to create a hyperlink for.
   b. Update the **primary_linked_field_name** and **secondary_linked_field_name** columns for the selected row with the value that you entered created in the DB view. For more information, see Example Data for Custom Field Tables.

Inactivating a Field in Persistence Database

You can inactivate a field if it is no longer required. An inactive field is not available in Navigator.

To inactivate a field:

- In the **navigator_entity_field_master** table in the **Persistence** database, set the value for the **delete_status** field to 1.

Adding Custom Action Button

You can add a custom action button to the Navigator Result Grid Action Menu to enable your users to perform specific actions on records that are returned from Navigator searches. You can add multiple custom action buttons for each Navigator search type and define the action for each button. You can also modify the properties of an existing action button.

To add a custom action button:

- Insert data in the Persistence database
- Create a new switch case to implement custom logic for the action
- Create a new UI resource for each custom entity and each action button

If you no longer require an action button, you can remove it from the action menu.

Inserting Data in Persistence Database

Read a description and see example values for each table that needs to be modified.

To insert data in the Persistence database:

1. Using Microsoft SQL Server 2008, access the **Persistence** database, right-click the **ActionListProfile** table and select **Edit Top 200 Rows**.
2. Scroll to the bottom of the displayed result grid, and insert one row for each new action button that you intend to add to the Navigator Result Grid Action Menu.
3. Repeat steps 1 and 2 for the following tables:
   - ActionListProfileML table
   - Broker table

- EndPoint table
- ActionButton table
- ActionButtonML table
- ActionGroup table - enter a row in this table only if you are creating a new action group.
- EntityGroupMapper table - enter a row in this table to assign a new action group to an existing or to a custom entity.

4. Create a UI resource for each custom entity.
5. Open the navigator_entity_master table, and enter the UI resource name in the GAMEntityResourceName column for the appropriate entity.

Creating New Switch Case

Create a new switch case in the JavaScript function to implement custom logic for the custom action button that you are creating.

To create a new switch case:

1. On the Onyx UI server, navigate to the **C:\Program Files\Onyx\EmployeePortal\QuickSearch** folder.
2. In an editor such as Notepad, open the **NavigatorSearchHost.asp** file.
3. Navigate to the JavaScript function **jsOpenGamAction**.
4. Add one switch case for each new action button that you created. For examples of a switch case, see the existing switch cases for the default action buttons in this file.

   The name that you enter for the switch case must be the same as the value you entered in the ActionName column of the ActionListProfile table.

5. Save your changes and close the **NavigatorSearchHost.asp** file.

Inactivating an Action Button

You can inactivate an action button if it is no longer required. An inactive action button is not available in Navigator Result Grid Action Menu.

To inactivate an action button:

- In the **ActionButton** table in the Persistence database, set the value for the **RecordStatus** field for the field to **0**.

Creating New UI Resource and Granting Permissions

UI resources enable you to define the actions that your users can or cannot perform. Onyx provides UI resources for all default UI elements, such as entities, fields, and actions buttons. For each custom UI element that you add to your Onyx system, you must create a new UI resource and grant the necessary permissions.

To create a new UI resource:

1. Log on to **OES Security Administration**, and expand **Resources**.
2. In the navigation area, select **UIResources** and click the **Add** button at the top left.
3. In the **Edit Resource** window, type the Resource ID in the specified format, and enter a Description to identify the resource.

   For reference, look for resources with Navigator in the name.
4. Click **Manage Permissions**.
5. In the navigation area expand **Roles**.
6. Select the following roles one by one, and click the **Add** button in the **Edit Resources** window: Administrator, OEP.administrator, OEP.user, and OEP.poweruser.
7. Click **Done** when you've finished adding the desired roles.
8. Click **Save**.
9. Update the relevant table in the Persistence database with the UI resource ID you created.

The tasks explained in this topic are:

- UI Resource Format
- Updating UI Resource in Persistence Database

UI Resource Format

When you create UI resource IDs for Navigator, we recommend that you follow the format: UI:OEP.<Entity_name.UI Element.Resource_Permission>, where:

- Entity_name is the name of the search type that the field is added to.
- UI Element is the UI resource name that you want to grant permissions for. This value is optional for some entity level UI resources.
- Resource_Permission is the permission for the resource.

Updating UI Resource in Persistence Database

After creating the UI resource IDs, you must update the appropriate tables in the Persistence database.

To do this, from the UI Resource ID, remove the text that specifies the permissions, and copy the remaining text to the ResourceName column in the appropriate table.

For example, if the new UI resource for a custom field is UI:OEP.SupportIncident.PriorityDesc.View, then enter 'UI:OEP.SupportIncident.PriorityDesc' in the ResourceName field in the navigator_entity_field_master table.

UI resource type	UI resource ID format	Resource name value to update in Persistence database	Table to update	Column to update
Search	UI:OEP.<Resource	UI:OEP.<Resource	navigator_	EntityResourceName

UI resource type	UI resource ID format	Resource name value to update in Persistence database	Table to update	Column to update
Type entity name	Name>.<Resource_ Permission>	Name>	entity_ master	
Search criteria field or result grid column	UI:OEP.<Resource Name>.<Resource_ Permission>	UI:OEP.<Resource Name>	navigator_ entity_field_ master	ResourceName
Result grid action menu	UI:OEP.<Resource Name>.<Resource_ Permission>	UI:OEP.<Resource Name>	navigator_ entity_ master	GAMEntityResourceName
Result grid action button	UI:OEP.<Resource Name>	UI:OEP.<Resource Name>	ActionButton	ResourceName

Supporting Multiple Languages

Navigator is supported on multilingual installations of Onyx in the following topology:

- Single OTDB
- Single OEAS with a single logical application with the default site ID
- A separate installation of OEP for each language

Warning: In the Navigator Results List, the Assigned To field for work tickets and tasks appears blank if the assigned user's language preference is different from the language of the website that you are logged on to. To view the Assigned To user, you must open the record.

To support multiple languages in Navigator, you must perform the following steps for each language that you want to support.

- Modify OTDB to support multiple languages
- Insert language-specific data in the Persistence database
- Localize Onyx Gateway Service error messages
- Localize script status and email priority domain data
- Update the configuration file for each website

Modify OTDB to support multiple languages

You must insert language-specific data into the OTDB in order to support multiple languages. For information on doing this, see **Customize Database (OTDB).**

Important: The language code that you create for each language is case-sensitive. Be sure to enter the language code throughout Onyx using the same case as in the OTDB Language table.

Insert language-specific data in the Persistence database

In the Persistence database, domain data is contained in two types of tables: language-independent tables, and language-dependent tables. You must update both these types of tables with data for each language that you intend to support.

To update the Persistence database tables:

1. Ensure that you have synchronized the OTDB and Persistence database so that the latest domain data is available in the Persistence database. For information on doing this, see the Onyx Installation Guide.
2. In each language-dependent table, create one row of domain data for each language code.

   For example, in the navigator_entity_master_ml table, create one row for 'Companies' with language code=ENG. Create another row for 'Société' with language code=FRA, and so on for each language code that you have created.

Localize Onyx Gateway Service error messages

To update error messages that are controlled through the Onyx Gateway Service , you must modify the NavigatorMessages.txt file with the language-appropriate text, and then create a DLL for each language. Before you begin, be sure to have a list of the culture names for each language in which you want to localize Navigator. Perform this process on a computer that has Microsoft Visual Studio 2010 installed.

Note: To find the culture name for a language, refer to http://msdn.microsoft.com/en-us/library/system.globalization.cultureinfo(v=vs.80).aspx

To create the resources DLL:

1. In your Onyx 7.8 installation package, navigate to **\Customization Support\CRArchitecture\CMRestGateway**, and copy the NavigatorMessages.txt file to a computer on which you have installed Microsoft Visual Studio 2010.
2. Open **.NET SDK Command Prompt** and perform the following steps:
   - Enter a command to point to the directory where you copied the NavigatorMessages.txt file.
   - Execute the following command, substituting the bracketed text with the appropriate values. This converts the NavigatorMessages.txt file into a .resources file.

```
resgen NavigatorMessages.txt NavigatorMessages.<culture-name>.resources
```

An example of the command: resgen NavigatorMessages.txt NavigatorMessages.ja-jp.resources

- Execute the following command, substituting the bracketed text with the appropriate values. This creates the required DLL.

```
al /t:lib /embed:NavigatorMessages.<culture-name>.resources
/culture:<culture-name> /out:NavigatorMessages.resources.dll
```

An example of the command: `>al /t:lib /embed:NavigatorMessages.ja-jp.resources /culture:ja-jp /out:NavigatorMessages.resources.dll`

3. On the **OEAS server computer**, navigate to `C:\Program Files\Onyx\AppServer\Applications\Onyx\OnyxGatewayService`
4. Create one folder for each language, and name it with the culture name, such as ja-jp for Japanese.

Note: When no language-specific dll is present, the culture-neutral dll is used to retrieve error messages..The default location for the culture-neutral dll is `C:\Program Files\Onyx\AppServer\Applications\Onyx\OnyxGatewayService\NavigatorMessages.dll`.

5. Copy the DLL for each language into the respective language folder.
6. In an editor such as Notepad, open the **CMGateway config.exe** file.
7. Find the text **Configuration\AppSettings**, and type text in the following format, substituting the bracketed text with the appropriate values.

```
<add key="<CM-Language-Code>" value="<.net-culture-name>"/>
```

An example of the text: `<add key="JPN" value="ja-jp"/>`

8. Save and close the configuration file.
9. To modify an error message, make the necessary modifications in the **NavigatorMessages.txt** file, recreate the DLL, and copy it to the appropriate language folder, following the steps listed above.

Important: Be sure to back up the modified .txt file. You will need it when you want to make further modifications and when you want to upgrade to a newer version of Onyx.

Localize Email Priority and Script Status domain data

You must also modify the XML files for Email Priority and Script Status to include domain data in each language that you want to support.

1. On the OEAS server, navigate to the app_data folder within your Onyx Gateway Service install folder. The default location for this folder is `C:\Program Files\Onyx\AppServer\Applications\Onyx\OnyxGatewayService\App_Data.`
2. Open the EmailPriorityDomainData.xml file, and add the domain data for Email Priority in each language that you want to support. The domain data in the default installation language is already present and does not need to be added.
3. After you've finished adding the data, save and close the file.
4. Open the ScriptStatusDomainData.xml file, and add the domain data for Script Status in each language that you want to support. The domain data in the default installation language is already present and does not need to be added.
5. After you've finished adding the data, save and close the file.

Update the configuration file

1. On the OEP server computer for each language installation, navigate to the Configuration folder within the OEP installation folder. The default location for this folder is `C:\Program Files\Onyx\EmployeePortal\CRArchitecture\Configuration.`
2. Using a text editor such as Notepad, open the Config.xml file.
3. In the node **add.baseAddress**, add the installation language code of the website in the following format: `"http://<Fully qualified ComputerName>:69/ServiceGateway/"language="languagecode"/>`

```
<services>
  <service name="CmService.Client">
    <host>
      <baseAddresses>
        <add
baseAddress="http://123.xyz.com:69/ServiceGateway/"language="ENG"/>
      </baseAddresses>
<WebAPIbaseAddresses> <add baseAddress="http://<OEP Server Name>/<OEP_
Website Name>/" language="ENG"/> </WebAPIbaseAddresses>
    </host>
  </service>
</services>
```

> **Important:** Be sure to enter the language code using the same case as in the OTDB Language table.

4. Save the Config.xml file.
5. On the OEP client computer, clear the temporary internet files and browser cache.

Performing Custom Operations Based on Navigator Queries

When users run queries on Navigator, the following metadata is sent to the OTDB:

- User Id
- Site Id
- Language
- Application Name
- Entity Id
- Session Id

You can use this information to perform further custom operations, based on your business needs. Some examples of custom operations are:

- Maintaining an audit log for all queries run
- Performing actions on linked database servers before executing a query
- Restricting certain types of data from being displayed, depending on the user profile
- Modifying 'where' conditions based on user role

To define the custom operations that you want to run, modify the opSearchQueryExecute stored procedure on the OTDB. Each time a user runs a Navigator query, the custom operations that you defined in the stored procedure are performed.

To define custom operations:

1. In SQL Server Management Studio, navigate to the OTDB, and go to Programmability>Stored Procedures.
2. Select dbo.opSearchQueryExecute, right-click, and select Script Stored Procedure as>ALTER To, and click New Query Editor Window.
3. Add the script that will perform your custom operation.
4. Execute the query.

Reference Values

Use this information to quickly find values for certain columns in the Persistence database.

- Entity IDs and entity views
- Data type
- Control type
- Field state
- Cache name
- Reserved field names
- Reserved entity names

- Example Data for Custom Entity Tables
- Example Data for Custom Field Tables
- Example Data for Custom Action Tables
- Example Data for Inline Editing Tables

Entity IDs and entity views

The following table lists the default entities available in Navigator, with the entity ID and the name of the respective entity view in the Onyx database.

In the entity view, each field included in the view is represented in the format [TableAlias].[field_name] AS [FieldAlias]. For example, c.company_id AS primaryId, where the field_name company_id from the Company table is displayed as the primary ID.

Entity ID	Entity Name	View Name
1	Company	company_navigator_view
2	Individual	individual_navigator_view
3	SalesOpportunity	incident_sales_navigator_view
4	ServiceRequest	incident_service_navigator_view
5	SupportIncident	incident_support_navigator_view
6	WorkTicket	workticket_navigator_view
7	Email	email_navigator_view
8	Task	task_navigator_view
9	Appointment	appointment_navigator_view
10	Script	script_navigator_view
11	Forecast	forecast_navigator_view
12	Product	product_navigator_view
13	Document	Document_navigator_view

Data type

The following table lists the data type options available for Navigator fields.

Data type	Description	Comment
BIT	True or false type of values	
CHAR255	Text values	
DATETIME	Date and time related	If the field appears in the Navigator Result Grid, meaning that the value in the use_type column for the field is 2, enter DateRange as the value in the

Data type	Description	Comment
	information	Control_Type column. This ensures that when the field is enabled for editing, the Date Picker control appears, allowing users to select a date. LONGTIME SHORTTIME SHORTDATE LONGDATE SHORTDATETIME LONGDATETIME SHORTDATELONGTIME LONGDATESHORTTIME
FLOAT	Decimal Values	If the field appears in the Navigator Result Grid, meaning that the value in the use_type column for the field is 2, enter Float as the value in the mask_ format column. This ensures that when the field is enabled for editing, the data is validated based on the requirements for the Float type.
INTEGER	Integer values	If the field appears in the Navigator Result Grid, meaning that the value in the use_type column for the field is 2, enter Integer as the value in the mask_format column. This ensures that when the field is enabled for editing, the data is validated based on the requirements for the Integer type.

Control type

The following table lists the control type options available for Navigator fields.

Control Type	Section	UI Control
Hidden	Search Criteria and Result Grid	Use this option for required fields that will not be visible in Navigator.
TextBox	Search Criteria and Result Grid	Use this option for Text box fields.
DateRange	Search Criteria and Result Grid	Use this option for date fields.
Tag Picker	Search Criteria and Result Grid	Use this option for the Tag field.
DropDown	Search Criteria	Use this option for combo box drop-downs in the Search Criteria panel.
CampaignPicker	Search Criteria	Use this option for the Campaign field.
ProductPicker	Search Criteria	Use this option for the Product field.
ScriptPicker	Search Criteria	Use this option for the Script field.
UserPicker	Search Criteria	Use this option for the User field.
ComboBox	Result Grid	Use this option for fields in which you can type or select from a list in the Result Grid window.
MaskEdit	Result Grid	Use this option for fields where you want to control how information should be displayed. Specify a mask in the 'mask_ format' field.
TreeSearch	Result Grid	Use this option for fields in which you can select values from a tree.

Field state

The following table lists the state ID for Navigator fields.

State ID	List box in Search Filter and Search Result Administration window
0	Fields containing unique information that distinguishes one record from another. Example: primaryId. This is the value for all fields that are set as Required fields in the corresponding Logical Business Object (LBO).
1	Disabled
2	Available
3	Selected
4	Result Grid Header field
5	Result Grid Detail field

Cache name

The following table lists the cache name for Navigator fields.

Cache Name	Bind To Control	UI Control
company.keyword	DropDown	Multi-select combo box
company.marketsector	DropDown	Multi-select combo box
company.sic	DropDown	Multi-select combo box
company.source	DropDown	Multi-select combo box
company.status	DropDown	Multi-select combo box
company.subtype	DropDown	Multi-select combo box
company.type	DropDown	Multi-select combo box
country	DropDown	Multi-select combo box
email.draft	DropDown	Multi-select combo box
email.followup	DropDown	Multi-select combo box
email.priority	DropDown	Multi-select combo box
email.sent	DropDown	Multi-select combo box
forecast.forecast.probability	DropDown	Multi-select combo box
gender	DropDown	Multi-select combo box
incident.sales.code1	DropDown	Multi-select combo box
incident.sales.code2	DropDown	Multi-select combo box

Cache Name	Bind To Control	UI Control
incident.sales.code3	DropDown	Multi-select combo box
incident.sales.code4	DropDown	Multi-select combo box
incident.sales.priority	DropDown	Multi-select combo box
incident.sales.source	DropDown	Multi-select combo box
incident.sales.status	DropDown	Multi-select combo box
incident.sales.type	DropDown	Multi-select combo box
incident.service.code1	DropDown	Multi-select combo box
incident.service.code2	DropDown	Multi-select combo box
incident.service.code3	DropDown	Multi-select combo box
incident.service.code4	DropDown	Multi-select combo box
incident.service.priority	DropDown	Multi-select combo box
incident.service.source	DropDown	Multi-select combo box
incident.service.status	DropDown	Multi-select combo box
incident.service.type	DropDown	Multi-select combo box
incident.status	DropDown	Multi-select combo box
incident.support.code1	DropDown	Multi-select combo box
incident.support.code2	DropDown	Multi-select combo box
incident.support.code3	DropDown	Multi-select combo box
incident.support.code4	DropDown	Multi-select combo box
incident.support.priority	DropDown	Multi-select combo box
incident.support.source	DropDown	Multi-select combo box
incident.support.status	DropDown	Multi-select combo box
incident.support.type	DropDown	Multi-select combo box
incident.workticket.code1	DropDown	Multi-select combo box
incident.workticket.code2	DropDown	Multi-select combo box
incident.workticket.code3	DropDown	Multi-select combo box
incident.workticket.code4	DropDown	Multi-select combo box
individual.department	DropDown	Multi-select combo box
individual.keyword	DropDown	Multi-select combo box
individual.source	DropDown	Multi-select combo box

Cache Name	Bind To Control	UI Control
individual.status	DropDown	Multi-select combo box
individual.title	DropDown	Multi-select combo box
product	Product picker	Multi-select tree picker
product.source	DropDown	Multi-select combo box
product.status	DropDown	Multi-select combo box
region	DropDown	Multi-select combo box
sales.keyword	DropDown	Multi-select combo box
sales.product	DropDown	Multi-select combo box
script	Script picker	Multi-select tree picker
script.status	DropDown	Multi-select combo box
service.keyword	DropDown	Multi-select combo box
service.product	DropDown	Multi-select combo box
support.keyword	DropDown	Multi-select combo box
support.product	DropDown	Multi-select tree picker
task.category	DropDown	Multi-select combo box
task.keyword	DropDown	Multi-select combo box
task.priority	DropDown	Multi-select combo box
task.status	DropDown	Multi-select combo box
task.type	DropDown	Multi-select combo box
trackingcode	Campaign picker	Multi-select Tree picker
users	DropDown	Multi-select combo box
workticket.keyword	DropDown	Multi-select tree picker
workticket.priority	DropDown	Multi-select combo box
workticket.product	DropDown	Multi-select combo box
workticket.severity	DropDown	Multi-select combo box
workticket.source	DropDown	Multi-select combo box
workticket.status	DropDown	Multi-select combo box
workticket.type	DropDown	Multi-select combo box

Reserved field names

The following table lists the values that you cannot use to refer to custom fields in the Onyx database view and in the field_name column of the navigator_entity_field_master table.

Reserved field names	Reserved field names	Reserved field names	Reserved field names
Append	folderId	moduleId	queryType
appName	folderName	navigatorEntity	remarks
authMode	group	objectTypeEnum	Rename
Begin	isBookmark	parameter	resourceId
categoryID	IsEditable	parentFolderId	Rollback
columns	ishomePageQuery	parentId	rowIndex
Commit	lang	Password	SequenceIndex
countryId	lboId	plpToken	siteID
Create	lboMethod	profileId	SortOrder
credential	listDescription	protectionMode	timeZoneOffset
Delete	listDomain	queryId	transactionHandle
DisplayName	listName	queryMode	transactionType
FieldID	MaxRecords	queryName	Update
FieldState	mode	queryParameter	userId

Reserved entity names

The following table lists the values that you cannot use for custom entities that you create.

Reserved Entity Names	Reserved Entity Names	Reserved Entity Names
ApplicationConfiguration	Forecast	Query
Appointment	GridActionMenu	QueryGroup
Authenticate	HelperMethod	Reminder
Common	Individual	RuntimeObjectBuilder
Company	JsonWriter	SalesOpportunity
Customer	LboConfiguration	Script
CustomerManagement	ModelClasses	SecurityPolicy
CustomerManagementExtended	OtmDispatcher	ServiceRequest
DomainData	OTMInteropException	ServiceRequestInspector
DomainDataSynchronizer	PersonalList	Session

Reserved Entity Names	Reserved Entity Names	Reserved Entity Names
Email	PrintEngine	SupportIncident
Entity	Product	Task
Fault	ProfileManager	WorkTicket

Example Data for Custom Entity Tables

- The navigator_entity_master table stores all the entity IDs that you intend to use in Navigator.

Example data for the navigator_entity_master table

Field Name	Field Description	Sample Field Value
entity_id	Primary ID – Enter the next sequential number after the last used value in this field. For example, if the last used value for entity_id is 12, then the new value should be 12 + 1 = 13 To find the last used value, run the following query on the Persistence database: SELECT max(entity_id) FROM navigator_entity_master	13
site_id	The Site ID that you use for Onyx.	1
entity_name	The value for the entity. This should be an alphanumeric value that begins with a letter (an alphabetic character). Do not use special characters or spaces in this value. If creating a new entity, do not use an existing entity name, or a reserved name. For a list of names that cannot be used for custom entities, see Reserved Entity Names.	Customers
entity_view_name	The name of the view created for the entity.	customer_navigator_view
use_type	Enter 2 to indicate the type of operation. Additional options are reserved for future use.	2
page_size	The page size supported in the Result Grid. The default value is 20.	20
max_records	The maximum number of search records displayed in the Result Grid.	300

Field Name	Field Description	Sample Field Value
	The default value is 300, and the maximum value is 999.	
SortOrder	Enter the sort order for the field, in the format <field_id>,<sort order>. Type the field_id for the selected field from the navigator_entity_field_ master table. 0 = ascending sort order 1 = descending sort order	271, 0
IsActive	Indicate if the record is active. True = Active, False = Inactive.	True
GAMEntityResourceName	Enter the UI resource name for the GAM. Refer to Create a new UI resource .	UI:OEP.customer.gam
LboObjectId	The LboObjectId from the LboObject table that corresponds to the Navigator entity. This enables inline editing for the entity.	12
EntityResourceName	Enter the UI resource name for the Entity. Refer to Create a new UI resource .	UI:OEP.Navigator.customer.retrieve

- The navigator_entity_master_ml table stores the entity display name that corresponds to each language that you've set up in Onyx.

Example data for the navigator_entity_master_ml table

Field Name	Field Description	Sample Field Value
entity_id	Enter the same value that you entered in the entity_id column of the navigator_entity_master table.	14
site_id	The Site ID that you use for Onyx.	1
Language_ code	Enter the language code based on your version of Onyx. Enter 'ENG' for English if you've installed the English version of Onyx. Enter 'JPN' for Japanese if you've installed the Japanese version of Onyx. Do not enter any other values for language code or unexpected results may occur.	ENG
entity_ display_ name	Enter the value that you want to use as the display name for the custom entity that you are adding.	Customers

Example Data for Custom Field Tables

- The navigator_entity_field_master table stores information for each field that is included in the entity view.

Example data for the navigator_entity_field_master table

Field Name	Field Description	Sample Field Value
field_id	Primary ID – Enter the next sequential number after the last used value in this field. For example, if the last used value for field_id is 617, then the new value should be 617 + 1 = 618 To find the last used value, run the following query on the Persistence database: SELECT max(field_id) FROM navigator_entity_field_master	618
site_id	The Site ID that you use for Onyx.	1
entity_id	Enter the entity ID that you are adding the field for. For a list of existing entity IDs, see Navigator Entities, or enter the entity ID for the custom entity that you created.	3
field_name	The value entered for the fieldalias in the Onyx database view for the field. For a particular entity, each field should contain a unique value in this column. There are some reserved field names, which cannot be used as field names for custom fields. For a list of reserved field names, see Reserved Field Names.	country
position	Enter the next sequential number after the last used value in this field. For example, if the last used value for position is 28, then the new value should be 28 + 1 = 29. To find the last used value, run the following query on the Persistence database: For Search Filters fields: SELECT max(position) FROM navigator_entity_field_master WHERE entity_id =<the value entered for entity_id field in this table> AND use_type = 0 For Result Grid fields: SELECT max(position) FROM navigator_entity_field_master WHERE entity_id =<the value entered for	29

Field Name	Field Description	Sample Field Value
	entity_id field in this table> AND use_ type = 2	
type	Enter the data type for the field. The value you should enter here depends on the value entered in the use type column, and it determines the values to enter in the control_type and mask_ format columns. For a list of field types, refer to Data Type.	CHAR255
parent	If the field is a child field, enter the corresponding parent field_name - For example, if the field is state, enter country as the parent. Otherwise, enter NULL.	NULL
sequence_ index	Enter the next sequential number after the last used value in this field. For example, if the last used value for sequence_index is 28, then the new value should be 28 + 1 = 29. To find the last used value, run the following query on the Persistence database: For Search Filters: SELECT max (sequence_index) FROM navigator_ entity_field_master WHERE entity_id ==<the value entered for entity_id field in this table> AND use_type = 0 For Result Grid Fields: SELECT max (sequence_index) FROM navigator_ entity_field_master WHERE entity_id ==<the value entered for entity_id field in this table> AND use_type = 2	29
control_type	Enter the control type for the field. The value you should enter here depends on the value entered in the type column. For a list of control types, refer to Control Type.	DropDown
mask_format	For Search Filter fields: use NULL. For Result Grid fields: Default value = NULL. To apply this value to Phone and Postal Code fields, ensure that Country is also included as a field for that entity. For Phone fields, enter Phone. For Postal Code, enter PostalCode. For all other fields, the value you should enter here depends	NULL

Field Name	Field Description	Sample Field Value
	on the Datatype value entered in the type column.	
is_transaction	If the field is required to perform an operation on the entity in Onyx, enter TRUE. For example: Primary ID. For other fields, enter FALSE.	TRUE
use_type	For Search Filter fields, enter 0. For Result Grid fields, enter 2. The value you enter here determines the values you enter in the type, control_type, and mask_format columns. For the Site ID and Primary ID fields, enter 0. For all other fields that are required in the LBO, enter 2.	0
field_state	Enter whether the field should be in the Available, Selected, or Disabled list in the Navigator Search Criteria or Navigator Result Grid Administration window. You can also specify if a field should be in the Header or Detail section of the Search Result window, as well as if a field should not be visible within Navigator. For the values corresponding to each state, see Field State.	2
is_batch_ update	Specify if the field is available during a batch_update. The batch update functionality is available only for the default Incident, Work Ticket, and Task entities. NULL = Default value 0 = No (not a batch update field) 1 = Yes	0
ResourceName	Enter the UI resource name. See Create a new UI resource .	UI:OEP.Navigator.SalesOpportunity.Country
cacheName	This field is required only when you select control_type as 'DropDown' or 'Any Tree Picker', otherwise enter NULL. See Cache Name.	incident.type
CacheParentId	This field is required only when the cacheName is shared between fields and if the value is determined based on the parent ID. For example: cacheName = incident.type CacheParentId: 1= Support 2= Service	1

Field Name	Field Description	Sample Field Value
	3= Sales	
delete_status	Indicates whether the field is available in Navigator. 0=Available 1=Not available	0
IsReadOnly	Indicates whether the field can be made available for inline editing. Set the value to 1 for fields that should never be edited by users. This makes the field non-editable even if UI resource permissions are granted by the administrator. 0=NotReadOnly 1=ReadOnly	0
is_editable	Reserved for future use	0
width	Reserved for future use	0
is_sorted	Reserved for future use	0
sort_order	Reserved for future use	NULL
linked_property	Use when adding a lookup field. Enter the field_name of the field that you want to use as the lookup field.	NULL
primary_ linked_field_ name	Use when adding a hyperlink to a column. Enter the field_name which has the value of object type enum.	Refer the field_name “ownerSecondaryId” for the primary_linked_field_name value.
secondary_ linked_field_ name	Use when adding a hyperlink to a column. Enter the field_name which has the value sub-category of the obect type.	Refer the field_name “ownerSecondaryId” for the secondary_linked_field_name value.

- The navigator_entity_field_master_ml table stores the field display name that corresponds to each language that you have set up in Onyx.

Example data for the navigator_entity_field_master_ml table

Field Name	Field Description	Sample Field Value
field_id	Enter the same value that you entered in the field_id column of navigator_ entity_field_master table.	618
Language_ code	Enter the language code based on your version of Onyx. Refer to the Language table in the Onyx database. Do not enter any other values for language code or unexpected results may occur.	ENG

Field Name	Field Description	Sample Field Value
site_id	The Site ID that you use for Onyx.	1
display_ name	Enter the value that you want as the Display Name for the field that you are adding.	Owner Country

Example Data for Custom Action Tables

- The ActionListProfile table stores the profile ID for the action, and associates this profile ID with the corresponding action in the legacy JavaScript code.

Example values for the ActionListProfile table

Field Name	Field Description	Sample Field Value
ProfileId	This field is automatically populated with the next sequential number for the table row.	12
SiteId	The Site ID that you use for Onyx.	1
ActionName	The name that identifies the custom action. Enter the same value that you enter for the new switch case to map the action button to the legacy code.	CloneCompany
IsSynchronous	Indicate whether the action is a synchronous call or an asynchronous call. At this time, only synchronous calls are supported. False =Synchronous, True = Asynchronous.	False
InsertBy	Enter the username that inserted the record.	sa
InsertDate	Enter the date on which the record is inserted.	2010/05/19
UpdateBy	Enter the username that last updated the record.	sa
UpdateDate	Enter the date on which the record is last updated.	2010/05/19
RecordStatus	Indicate if the record is active. True = Active, False = Inactive.	True

- The ActionListProfileMl table associates each profile ID in the ActionListProfile table with a display name. This name is used to identify the action in the Action Button Configuration UI, which will be developed at a future date.

Example values for the ActionListProfileMl table

Field Name	Field Description	Sample Field Value
ProfileId	The profile ID that you are entering a display name for.	12

Field Name	Field Description	Sample Field Value
	To find the profile ID for an action, run the following query on the Persistence database, where <Name of the custom action> is the value for the ActionName column in the ActionListProfile table. Select ProfileId from ActionListProfile where ActionName='<Name of the custom action>	
SiteId	The Site ID that you use for Onyx.	1
Language_code	Enter the language code based on your version of Onyx. Refer to the Language table in the Onyx database. Do not enter any other values for language code or unexpected results may occur.	ENG
ActionDisplayName	Enter the text to display for and identify the action in the Action Button Configuration UI, which will be developed at a future date.	Clone company

- The Broker table associates each profile ID in the ActionListProfile table with a broker ID that initiates an action. At this time, it is possible to associate only one broker ID with each profile ID.

Example values for the Broker table

Field Name	Field Description	Sample Field Value
BrokerId	This field is automatically populated with the next sequential number for the table row.	23
ProfileId	The Profile id to associate the broker ID with. To find the profile ID for an action, run the following query on the Persistence database, where <Name of the custom action> is the value for the ActionName column. : Select profileId from ActionListProfile where ActionName='<Name of the custom action>'	12
SiteId	The Site ID that you use for Onyx.	1
Priority	The priority in which to run the broker. At this time, this value should always be 1.	1
InsertBy	Enter the username that inserted the record.	sa
InsertDate	Enter the date on which the record is inserted.	2010/05/19
UpdateBy	Enter the username that last updated the record.	?
UpdateDate	Enter the date on which the record is last updated.	2010/05/19
RecordStatus	Indicate if the record is active. True = Active, False = Inactive.	True

- The EndPoint table associates each broker ID to an endpoint ID that determines the destination for the action. At this time, only LegacyCodeEndPoint which opens an existing Onyx page, is supported.

Example values for the EndPoint table

Field Name	Field Description	Sample Field Value
EndPointId	This field is automatically populated with the next sequential number for the table row.	23
SiteId	The Site ID that you use for Onyx.	1
BrokerId	The BrokerID to associate the endpoint with.	23
Priority	The priority in which to call the end point. At this time, this value should always be 1.	1
EndPoint	The target URI that must be invoked by the action button. At this time, this field is not used and is reserved for future use.	NULL
EndPointTypeId	The provider type of the endpoint which handles invocation of the endpoint. At this time, the only supported value is 4, which represents LegacyCodeEndpoint.	4
InsertBy	Enter the username that inserted the record.	sa
InsertDate	Enter the date on which the record is inserted.	2010/05/19
UpdateBy	Enter the username that last updated the record.	sa
UpdateDate	Enter the date on which the record is last updated.	2010/05/19
RecordStatus	Indicate if the record is active. True = Active, False = Inactive.	True

- The ActionButton table associates each profile ID in the ActionListProfile table with an ActionButton ID. In this table, you specify the entity and group that the action belongs to, and the icon, sequence number, and resource ID to use for the button.

Example values for the ActionButton table

Field Name	Field Description	Sample Field Value
ActionButtonId	This field is automatically populated with the next sequential number for the table row.	12
SiteId	The Site ID that you use for Onyx.	1
ProfileId	The ProfileId to associate the action button ID with. To find the profile ID for an action, run the following query on the Persistence database, where <Name of the custom action> is the value for the ActionName column. : Select profileId from ActionListProfile where ActionName='<Name of the custom action>'	71

Field Name	Field Description	Sample Field Value
EntityId	The entity ID to associate the action button with. To find the entity ID corresponding to the entity name that you are creating the action button for, see the navigator_ entity_master table in the Persistence database.	1
GroupId	The Action Group ID in which to include the button. To find the action group ID, see the ActionGroup table in the Persistence database.	1
SequenceIndex	The sequence of the button within the selected action group.	2
Icon	The icon for the button in the Navigator Result Grid Action Menu. This icon will be mapped as relative to the folders present in employeeportal\quicksearch\clientbin folder. Ensure that the image format is Portable Network Graphics (png) and that the image size is 18*18.	Icon/AddContact.png
ResourceName	The UI resource to associate with the action button. To create a UI Resource, see create a new UI resource.	UI:OEP.Navigator.Company. CompanyCloneItem
InsertBy	Enter the user name that inserted the record.	sa
InsertDate	Enter the date on which the record is inserted.	2010/05/19
UpdateBy	Enter the username that last updated the record.	sa
UpdateDate	Enter the date on which the record is last updated.	2010/05/19
RecordStatus	Indicate if the record is active. True = Active, False = Inactive.	True

- The ActionButtonMl table associates each ActionButtonID in the ActionButton table with a tool tip that appears when a user points to the button.

Example values for the ActionButtonMl table

Field Name	Field Description	Sample Field Value
ActionButtonId	The action button ID to create the caption for. Enter the appropriate ActionButtonID value from the ActionButton table.	12
SiteId	The Site ID that you use for Onyx.	1
LanguageCode	Enter the language code based on your version of Onyx. Refer to the Language table in the Onyx database. Do not enter any other values for language code or unexpected results may occur.	ENG
Caption	The tool tip to display when a user points to the	Clone this company

Field Name	Field Description	Sample Field Value
	button on the Navigator Result Grid Action Menu.	

- The ActionGroup table stores information for all the action groups that are created in Navigator. Enter a value here only if you are creating a new action group.

Example values for the ActionGroup table

Field Name	Field Description	Sample Field Value
GroupId	This field is automatically populated with the next sequential number for the table row.	3
SiteId	The Site ID that you use for Onyx.	1
GroupName	Enter a name for the action group.	ListAction
InsertBy	Enter the user name that inserted the record.	sa
InsertDate	Enter the date on which the record is inserted.	2010/05/19
UpdateBy	Enter the username that last updated the record.	sa
UpdateDate	Enter the date on which the record is last updated.	2010/05/19
RecordStatus	Indicate if the record is active. True = Active, False = Inactive.	True

- The EntityGroupMapper table maps action groups to each entity and stores information about the display sequence. Enter a value to assign a newly created action group to each entity, or to assign an existing action group to a custom entity.

Example values for the EntityGroupMapper table

Field Name	Field Description	Sample Field Value
EntityGroupMapperId	This field is automatically populated with the next sequential number for the table row.	25
SiteId	The Site ID that you use for Onyx.	
GroupId	The Action Group ID to include for the entity. To find the action group ID, see the ActionGroup table in the Persistence database.	
EntityId	The entity ID to associate the group with. To find the entity ID corresponding to the entity name that you are mapping the group to, see the navigator_entity_master table in the Persistence database.	
SequenceIndex	The sequence in which to display the group within the action menu.	

Field Name	Field Description	Sample Field Value
RowNumber	The row in the action menu in which to display the group.	
InsertBy	Enter the user name that inserted the record.	sa
InsertDate	Enter the date on which the record is inserted.	2010/05/19
UpdateBy	Enter the username that last updated the record.	sa
UpdateDate	Enter the date on which the record is last updated.	2010/05/19
RecordStatus	Indicate if the record is active. True = Active, False = Inactive.	True

Example Data for Inline Editing Tables

- The LboObject table stores the Logical Business Object (Lbo) names that inline editing is enabled for, and the method that is supported for each object. For each row that you add to this table, you must add at least one corresponding row in the LboPropertyMapper table.

Example values for the LboObject table

Field Name	Field Description	Sample Field Value
LboObjectID	This field is automatically populated with the next sequential number for the table row. Enter this value in the LboObjectId column of the navigator_entity_master table for the corresponding Navigator entity.	12
SiteId	The Site ID that you use for Onyx.	1
ObjectName	The value of the object name from Object Designer.	Company
MethodName	The method to use for the object. Currently, only the Update method is supported.	Update
InsertBy	Enter the username that inserted the record.	sa
InsertDate	Enter the date on which the record is inserted.	2010/05/19
UpdateBy	Enter the username that last updated the record.	sa
UpdateDate	Enter the date on which the record is last updated.	2010/05/19
RecordStatus	Indicate if the record is active. True = Active, False = Inactive.	True

- The LboPropertyMapper table associates each Logical Business Object (LBO) property with its corresponding field in the navigator_entity_field_master table.

Example values for the LboPropertyMapper table

Field Name	Field Description	Sample Field Value
LboPropertyId	This field is automatically populated with the next sequential number for the table row.	12
SiteId	The site ID that you use for Onyx.	1
FieldId	The value corresponding to the field in the Field id column of the navigator_entity_field_master table, that you plan to use as the lookup field.	618
LboProperty	The Lbo property in the OEAS corresponding to the Navigator field.	
LboObjectId	The LboObjectid from the LboObject table that corresponds to the Navigator entity that the field belongs to.	12
PropertyPath	The XPath that the property is located under in the Update Method XML. The path must end with / and must not contain the property. Currently, child objects are not supported for inline editing.	company/
InsertBy	Enter the username that inserted the record.	sa
InsertDate	Enter the date on which the record is inserted.	2010/05/19
UpdateBy	Enter the username that last updated the record.	sa
UpdateDate	Enter the date on which the record is last updated.	2010/05/19
RecordStatus	Indicate of the record is active. True=Active, False=Inactive.	True

Configuring Bubble MessageTimeout Value

To configure the bubble message timeout value, perform the following steps:

1. In the OEP server, browse to the `\Program Files\Onyx\EmployeePortal\CRArchitecture\Configurations` folder.
2. Open the `config.xml` file as administrator.
3. Enter the delay value under `<bubblePopUpWindow>` section.

Example:

```
<bubblePopUpWindow>
<error isEnable="false" delay="2" />
<information isEnable="true" delay="10" />
<warning isEnable="true" delay="7" />
<question isEnable="false" delay="5" />
</bubblePopUpWindow>
```

Where:

`isEnable="true"` enables the bubble popup window.

`isEnable="false"` disables the bubble popup window.

delay specifies the no. of seconds the bubble popup window is displayed.

4. Save the `config.xml` file.

Navigator Search Types

The Navigator search works in two ways:

- View
- Stored Procedure

The Stored Procedure search is introduced from Onyx 7.8. The stored procedure search is available for all entities, but out of the box the stored procedure is available only for the incident search. If you want to search other entities using Stored Procedures, you need to create your custom Stored Procedures for the intended entities.

Enabling the Stored Procedure or View Search

To enable the Stored Procedure or View Search

1. Connect to Persistence database.
2. Open **Navigator_entity_master** table.
3. Set the **SearchExecuteMode** column value to **1** to search using Stored Procedure or **0** to search using View for the entities based on your organization's preferences.

Example

The following script is an example of enabling the Stored Procedure for Sales Opportunity:

```
Update Navigator_entity_master set SearchExecuteMode=1 where entity_
name='SalesOpportunity'
```

Important: The parameters and naming conventions passed to the Stored Procedures should match the Views and vice versa.

Note: To add or remove the fields, contact your system administrator.

6

Onyx Insight

Navigator or Insight Detail View and Detail List View Customization

To customize the Detail View and Detail List View in the Navigator or Insight, do the following changes in Persistence database:

Adding Detail View Pane

The Detail View pane displays the primary details of the record selected on the Search Result Summary pane.

Following tables are used to add the Detail View pane:

- DetailView
- DetailViewGroup
- DetailViewGroupMl
- DetailViewField
- DetailViewFieldMl

DetailView

The entities like Company, Individual, Sales incidents etc., are defined here.

Following are the column configurations of the DetailView table.

Column	Description
PrimaryDataLbo	Mention Lbo object for the entity.
PrimaryDataLboMethod	Mention retrieve Lbo method name of the Lbo object.
SecondaryDataLbo / SecondaryDataLboMetho	Any more information that is not present in the result set of PrimaryDataLboMethod can be retrieved from this object method.
SaveLboMethod	Mention the method name that updates the data.

DetailViewGroup

Create groups specific to each detailview and they act as place holders for fields like name, secondary id, email id etc. It also contains collection data like address. Following are the column configurations of the DetailViewGroup table.

Column	Description
SequenceIndex	Specify sequence number to arrange the group order.
IsCollection	Set to 1 if the group contains collection data, otherwise set to 0.

Column	Description
	Note: Collection and non-collection data cannot be grouped together. **Example:** Configure all the address under one group, similarly email in another and so on. Non-collection fields like Company id or company name cannot be grouped with address or email.
Display only primary	In case of collection data, to display only primary details, set this column to 1, otherwise 0.
UIResource	Mention the UI resource permission if the display of the group needs to block or grant based on UI Resource Permission.

DetailViewGroupMl

This table contains language code and corresponding display name, which is used to display as the respective group header on the detail view form.

DetailViewField

Define the fields (Company, Individual, Secondary Id etc.,) that you need to display for this entity on Detail View pane.

Following are the column configurations of the DetailViewField table.

Column	Description
DetailViewId	Primary key ID of DetailView table.
GroupId	The group in which the field should be shown with.
LboProperty	Mention the property name that is specified in respective Lbo method.
GroupId	The group in which the field should be shown with.
DomainDataName	Specify domain data name if the data needs to be fetched from domain data. **Example:** “individual.type” or “individual.subtype” etc.
DomainDataParentId	Mention the domain data parent id for the field, if domain data is required.
ParentFieldId	Sepcify domain data parent ID, if any.
MaskFormat	Specify if any masking needs to be performed on the retrived Lbo data.
DataType	Specify the datatype of the field.
ControlType	For read only fields set this column as Label, otherwise choose the required control name. **Note:** If control type is not specified as Label, the same field is considered as editable. This is the only method to make a

Column	Description
	 field editable and update the value in edit mode.
IsRequired	
SequenceIndex	Choose the number that specifies the order where you want to see this field on the UI.
UIResource	Mention the UI resource permission if the display of the field needs to block or grant based on a UI Resource Permission.

DetailViewFieldMl

This table contains language code and corresponding display name, which is used to display as the respective field label in the detail view form.

Adding Detail List View Pane

The Detail List View pane displays more information of the selected result, like all the tasks for a customer and open incidents. It can be customized to get more info by altering the respective Lbo method and stored procedures.

Following tables are used to configure the Detail List View pane:

- DetailListViewForm
- DetailListView
- DetailListViewMl
- DetailListViewGroup
- DetailListViewGroupMl
- DetailListViewColumn
- DetailListViewColumnMl

DetailListViewForm

This table creates a logical form, which is unique for each entity.

Following are the column configurations of the DetailListViewForm table.

Column	Description
UIResource	Mention the UI Resource Permission if the display of the detail list view needs to block or grant based on UI Resource Permission.
DetailViewId	Primary key ID of DetailView table.

DetailListView

Following are the column configurations of the DetailListView table.

Column	Description
DetailListViewId	Primary key ID for the detail list view.
FormId	Primary ID of DetailListViewForm to link to a particular entity.
GroupId	The group in which the field should be shown with.
SequenceIndex	Choose the number that specifies the order where you want to see this particular list view on the UI.
DefaultSortColumn	Mention the column name to perform sort by default before displaying on UI.
RetrieveLBO	Specify Lbo object to which list view belongs to. **Example:** If task information is appears on this list view, mention task (object name).
RetrieveMethod	Specify retrievelist Lbo method name of the object mentioned.
RetrieveLboParam	Mention the method parameters for the Lbo method mentioned above.
UIResource	Mention the UI Resource Permission if the display of the detail list view needs to block or grant based on UI Resource Permission.

DetailListViewMl

The description column of the table makes the heading of the group.

DetailListViewGroup

Following are the column configurations of the DetailListViewGroup table.

Column	Description
GroupId	Primary ID of the group.
FormId	Primary ID of DetailListViewForm to link to particular entity.
UIResource	Mention the UI Resource Permission if the dispaly of the group needs to block or grant based on UI Resource Permission.

DetailListViewGroupMl

The description column of the table is the name of the group.

DetailListViewColumn

Following are the column configurations of the DetailListViewColumn table.

Column	Description
DetailListViewColumnID	Primary key of the table.
DetailListViewID	Primary key ID for the detail list view table.
LboProperty	Name of the property from the corresponding Lbo method.
Datatype	Datatype of the property and field.
Sequenceindex	Choose the number that specifies the order where you want to see this particular column on the UI.
MaskFormat	To apply masking on the property before showing on UI.

DetailListViewColumnMl

The description column of the table makes the header of the grid view column.

Note

To bring the **Detail View** and **Detail List View** in OEP for the newly created custom entity, add `DetailViewId` from **DetailView** table to the view created in standalone product crawler view as shown below:

```
CREATE VIEW [dbo].[incident_sales_insight_data_query_view] AS
SELECT
inc.incident_id AS search_external_id,
inc.incident_id AS attr_primary_id,
inc.secondary_id AS field_secondary_id,
'5' as attr_detailformid,
inc.private_access AS attr_private_access,
FROM incident
```

In the above example `attr_detailformid` is the **DetailViewId** from **DetailView** table in OEDB. For more information on views used in product crawler, see *Entity IDs and entity views on page 5-28*.

Granting or Denying UI Resource Permissions

Granting or denying of UI Resource Permissions for Details, Groups, Fields, and Lists on detail view is explained in this section.

Granting or Denying UI Resource Permission for Details

On Persistence DB, update the **UIResource** column value of the **DetailView** table to grant or deny the UI Resource Permission for the detail view. The UI Resource Permission should be created through the security administration from OES.

Example

To grant or deny detail view for Individual

1. Go to security administration on OES.
2. Navigate to **ResourcesUI** > **Resources** and click **Add**.
3. Provide the **Resource ID** and **Description**.
4. Click **Manage Permissions** and add a user for that permission (you can either grant or deny the permission for that user).
5. Perform **Com+Shutdown**.
6. Open **PersistenceDB** and navigate to the **DetailView** table.
7. Open **DetailView** table in edit mode.
8. Filter the results based on **DetailViewId**.
9. Update the **UIResource** field value with the newly created **UIResource**.

Note: Restart the OEP after updating the **UIResource** column values in the database to verify the changes.

Granting or Denying UI Resource Permission for Groups

On Persistence DB, update the **UIResource** column value of the **DetailViewGroup** table to grant or deny the UI Resource Permission for the groups available on the detail view. The UI Resource Permission should be created through the security administration from OES.

Example

To grant or deny groups available on individual detail view

1. Go to security administration on OES.
2. Navigate to **Resources** > **UI Resources** and click **Add**.
3. Provide the **Resource ID** and **Description**.
4. Click **Manage Permissions** and add a user for that permission (you can either grant or deny the permission for that user).
5. Perform **Com+Shutdown**.
6. Open **PersistanceDB** and navigate to the **DetailViewGroup** table.
7. Open **DetailViewGroup** table in edit mode.
8. Filter the results based on **GroupId**.
9. Update the **UIResource** field value with the newly created **UIResource**.

Note: Restart the OEP after updating the **UIResource** column values in the database to verify the changes.

Granting or Denying UI Resource Permission for Fields

On Persistence DB, update the **UIResource** column value of the **DetailViewField** table to grant or deny the UI Resource Permission for the fields available on the detail view. The UI Resource Permission should be created through the security administration from OES.

Example

To grant or deny fields available on individual detail view

1. Go to security administration on OES.
2. Navigate to **Resources** > **UI Resources** and click **Add**.
3. Provide the **Resource ID** and **Description**.
4. Click **Manage Permissions** and add a user for that permission (you can either grant or deny the permission for that user).
5. Perform **Com+Shutdown**.
6. Open **PersistanceDB** and navigate to the **DetailViewField** table.
7. Open **DetailViewField** table in edit mode.
8. Filter the results based on **FieldId**.
9. Update the **UIResource** field value with the newly created **UIResource**.

Note: Restart the OEP after updating the **[UIResource]** column values in the database to verify the changes.

Granting or Denying UI Resource Permission for Lists

On Persistence DB, update the **UIResource** column value of the **DetailListView** table to grant or deny the UI Resource Permission for the lists available on the detail view. The UI Resource Permission should be created through the security administration from OES.

Example

To grant or deny lists available on individual detail view

1. Go to security administration on OES.
2. Navigate to **Resources** > **UI Resources** and click **Add**.
3. Provide the **Resource ID** and **Description**.
4. Click **Manage Permissions** and add a user for that permission (you can either grant or deny the permission for that user).
5. Perform **Com+Shutdown**.
6. Open **PersistanceDB** and navigate to the **DetailListView** table.
7. Open **DetailListView** table in edit mode.

8. Filter the results based on **ListId**.
9. Update the **UIResource** field value with the newly created **UIResource**.

Note: Restart the OEP after updating the **UIResource** column values in the database to verify the changes.

Changing the Sequence

changing the sequence of Groups, Fields, and Lists is explained in this section.

Changing the Sequence of Groups

On Persistence DB, update the **SequenceIndex** column values in the **DetailViewGroup** table to change the sequence of groups present on the Detail and Edit forms.

Example

To change the sequence of groups on Individual Detail or Edit form

1. Open **PersistanceDB** and navigate to the **DetailViewGroup** table.
2. Open **DetailViewGroup** table in edit mode.
3. Filter the results based on **DetailViewId** i.e., DetailViewId=1 is for Individual.
4. Update the **SequenceIndex** column values.

Note: Restart the OEP after changing the **SequenceIndex]** column values in the database to verify the changes.

Changing the Sequence of Fields in a Group

On Persistence DB, update the **SequenceIndex** column values in the **DetailViewField** table to change the sequence of groups present on the Detail and Edit forms.

Example

To change the sequence of fields on Individual Detail or Edit form

1. Open **PersistanceDB** and navigate to the **DetailViewField** table.
2. Open **DetailViewField** table in edit mode.
3. Filter the results based on **GroupId** i.e., GroupId=1 is for Individual.
4. Update the **SequenceIndex** column values.

Note: Restart the OEP after changing the **SequenceIndex** column values in the database to verify the changes.

Changing the Sequence of Lists on List View

On Persistence DB, update the **SequenceIndex** column values in the **DetailListView** table to change the sequence of lists present on the Detail and Edit forms.

Example

To change the sequence of lists on Individual Detail or Edit form

1. Open **PersistanceDB** and navigate to the **DetailListView** table.
2. Open **DetailListView** table in edit mode.
3. Filter the results based on **GroupView** i.e., GroupView=1 is for Individual.
4. Update the **SequenceIndex** column values.

Note: Restart the OEP after changing the **SequenceIndex** column values in the database to verify the changes.

Masking

The mask format is supported for the Phone Number, Postal Code, Numbers, Currency, and Date and Time.

On persistence DB, update the following formats in the **MaskFormat** column values in the **DetailViewField** tablefor the respective fields based on the FieldName.

Phone Number

Following is the mask format for phone:

```
Mask: PhoneMask
```

Postal Code

Following is the mask format for postal code:

```
Mask: PostalMask
```

Numbers

Following is the mask formats for numbers:

- `Integer: (Data Type: int) – '0000'`
- `Float: (Data Type: float) – '0000.00'`
- `Decimal: (Data Type: decimal) – '0000.0000'`

Date and Time

Following are the mask formats for Date and Time:

SUPPORTED FORMAT SPECIFIER	USER INTERFACE
h:mm A	2:00 AM
h:mm:ss A	2:00:30 AM
DD-MMM-YYYY	10-Jan-2013
dddd, MMMM DD, YYYY	Thursday, January 10, 2013
DD-MMM-YYYY h:mm A	10-Jan-2013 2:00 AM
dddd, MMMM DD, YYYY h:mm:ss A	Thursday, January 10, 2013 2:00:30 AM
dddd, MMMM DD, YYYY h:mm A	Thursday, January 10, 2013 2:00 AM
DD-MMM-YYYY h:mm:ss A	10-Jan-2013 2:00:30 AM
YYYY-MM-DDTHH:mm:ss	2013-01-10T02:00:30
YYYY-MM-DD HH:mm:ss	2013-01-10 02:00:30
MM-DD-YYYY	01-10-2013
DD-MM-YYYY	10-01-2013
MM/DD/YYYY	01/10/2013
DD/MM/YYYY	10/01/2013

Making Fields Editable

On Persistence DB, update the **IsEditable** column values in the **DetailView** table to make the fields editable or non-editable on the Detail View Pane.

Example

To update IsEditable property of the Individual entity

1. Open PersistanceDB and navigate to the **DetailView** table.
2. Open **DetailView** table in edit mode.
3. Filter the results based on **DetailView** i.e., DetailViewId=1 is for Individual.
4. Update the **IsEditable** column values.

Important: To make a particular field editable, update the **ControlType** column to appropriate control.
Example: `TextField` for text editable field and `DatePicker` for date time field.

Note: Restart the OEP after updating the **IsEditable** column values in the database to verify the changes.

Control Types

Following are the control types, which can be editable on the edit form:

- DatePicker
- DropDown
- Hyperlink
- NumberField
- TextField
- UserPicker
- ComboBox
- DateTimePicker
- TagPicker
- TextArea
- TimePicker
- TreePicker

7

Mobile Bookmarks

Overview

Mobile bookmarks are queries created in Navigator that you can view in Onyx Mobile. While creating a Navigator query, users can mark it as a Mobile Bookmark. When users log on to Onyx through their iOS or Android device, these bookmarks are listed as menu items on the Home Page.

The topics explained are:

Navigator Queries as Mobile Bookmarks

When creating Navigator queries, users can now save them as personal mobile bookmarks. Currently, users can create mobile bookmarks for the following Navigator entities:

- Companies
- Individuals
- Sales Opportunities
- Service Requests
- Support Incidents
- Tasks
- Products

Warning: To save an existing query as a mobile bookmark, users must select the Subscribe as Mobile Bookmark button for the query in the Saved Queries section.

User-specific Mobile Bookmarks

In addition to mobile bookmarks created from Navigator, you can modify and run the opInsertOOBBookMark.sql script included in the Installation Package on the Onyx Persistence database (OPDB) to create the following user-specific mobile bookmarks:

- Incidents assigned to me - opens a list page with all sales, support, and service incidents assigned to the logged-on user. To create this bookmark, you must first create a new Navigator entity named Incidents. For information on doing this, see Creating Incidents as a custom entity.
- Tasks assigned to me - opens a list page with all the tasks assigned to the logged on user.

To create user-specific mobile bookmarks:

1. In your Onyx 7.8 Installation Package, navigate to Customization Support>Mobile Bookmarks.
2. Copy the opInsertOOBBookMark.sql file to your Onyx database server and execute it on the Onyx Persistence database. This creates the opInsertOOBBookMark procedure in the Persistence database.
3. On the Onyx Persistence database, execute the following query, replacing OEDBDEMO with the name of your Onyx Transaction database. This creates mobile bookmarks for all users in Onyx, other than the 'sa' user.

```
BEGIN

BEGIN TRANSACTION

EXEC opInsertOOBBookMark 'OEDBDEMO'

If @@error >0

ROLLBACK TRANSACTION

Else

COMMIT TRANSACTION

END
```

4. Log on to Onyx Mobile as a user other than the 'sa' user, and verify that the following bookmarks are available on the Home Page:
   - My Incidents
   - My Tasks

   Note: If you have not created Incidents as a mobile bookmark, then no bookmarks for Incidents will be created.
5. When you add new users in Onyx, run the script mentioned in step 3 to create mobile bookmarks for the new user. Re-running the script will not create duplicate entries for existing users.

Creating Incidents as a custom entity

To create user-specific bookmarks for Incidents, you must create a new Navigator entity called Incidents in the Persistence database.

To do this, perform the following steps:

1. In your Onyx Installation Package, navigate to Customization Support>Mobile Bookmark, and copy the following scripts to your Onyx database server:
   - incident_navigator_view.sql
   - insertIncidentUIResoure.sql
   - insertIncidentCustomentity.sql
2. Review and modify these scripts to match your installation of Onyx CRM and your business needs.
3. On the Onyx Transaction database, run the following scripts:
   - incident_navigator_view.sql
   - insertIncidentUIResoure.sql
4. On the Onyx Persistence database, run the following script:
   - insertIncidentCustomentity.sql

   This creates the required entries and views in the databases for a new custom entity: Incidents.
5. Use the OGS Custom Entity Helper tool to create the custom entity DLL in OGS. For information on doing this, see Administering Navigator.

Configuring List Screens for Mobile Bookmarks

When users tap a mobile bookmark on the Home Page, a list screen displays all the records that meet the search criteria specified for the bookmark. Use the list profile XML file to specify which fields are displayed on the list screen for each entity.

The fields that you select to display on the mobile bookmark list page should already be configured in the Navigator entity view that is created for the entity. You can customize these views to add new fields. For more information on adding custom fields to Navigator entities, see the relevant section in the Onyx Technical Guide.

1. In the Onyx Transaction Database, open the Navigator Entity View corresponding to the entity for which you want to configure mobile bookmarks.
2. Note the values for field_name corresponding to the fields that you want to display on the mobile bookmark list screen. The value in this field is the name of the column in the Onyx Transaction Database from which to retrieve data.

3. On the OEAS server, navigate to the Onyx Gateway Service installation folder, and open the list profile XML file for the entity. The default file path is `C:\Program Files\Onyx\AppServer\Applications\Onyx\OnyxGatewayService\App_Data\Profiles`.
4. In the XML file, navigate to the field that you want to display on the list screen. To add a row for a new field, see Configuring a List Screen.
5. In the ViewColumnName attribute for the field, enter the value used for field_name corresponding to the field that you noted in step 2.
6. Leave this attribute blank for a field that you do not want to display on the mobile bookmark list screen.
7. Repeat the above process for each field that you want to display on the list screen.
8. Save and close the XML file.
9. Repeat this process for each entity that users can create bookmarks for.
10. For users to be able to save a Navigator query as a mobile bookmark, the mobile-specific fields should also be selected in the Navigator Result Grid. Therefore, do one of the following:
    - In Navigator, include the mobile-specific fields as default fields in the Header or Detail columns for the entity so that they are automatically included in the Navigator Result Grid when users save a query.
    - Alternatively, leave these fields in the Available column, so that users can select them when creating queries for mobile bookmarks.

It is recommended that when you change these fields for an entity, you inform users about the changes made so that they can modify their mobile bookmarks.

Onyx Mobile Details Screens

The Onyx Mobile Details screens are read-only screens that display essential information for a specific record. They provide links to view, add, and modify information about the record.

Company Details Screen

The Company Details screen includes name, address, phone, email, and primary contact information for company records. It also provides links to view company contacts and incidents. Users can call or email a company, edit the company record, add a company to their favorites, view directions to the company's primary address, and also add new incidents for the company.

Individual Details Screen

The Individual Details screen includes name, address, phone, email, and primary contact information for individual records. It also provides links to view individual contacts and incidents. Users can call or email an individual, edit the individual record, add an individual to their favorites, view directions to the individual's primary address, and also add new incidents for the individual.

Incident Details Screen (Sales/Service/Support)

The Incident Details screen includes information such as description, ID, owner, priority, status, and type details for incident records. It also displays the last work note added for the incident. Users can edit the incident record, view contacts, tasks, and work notes for the incident as well as add new work notes or tasks.

Task Details Screen (Sales/Service/Support)

The Task Details screen includes information such as description, ID, owner, priority, status, and type details for task records. It also displays the last work note added for the task. Users can view and add work notes to the task.

The tasks explained in this topic are:

- Configuring Details Screen
- Understanding Tap Type Values
- Understanding Action Values
- Understanding Context Variable Values

Configuring Details Screens

The fields and actions available on each Details screen depend on the configuration within the XML file that corresponds to the screen. Use this information to configure XML files for Details screens.

Note: Values you enter in the XML file are case-sensitive.

To configure Details screens:

1. Open the XML file corresponding to the Details screen that you plan to modify. For a list of XML files and the screens that they correspond to, see About XML files.
2. Configure attributes for each element within the XML file.
3. Configure attributes for each group within the XML file. You can add or remove groups on this screen.
4. Configure attributes for each field within each group.
5. Configure attributes for each item in the Do More group. This group contains navigation menu items that open other Onyx Mobile screens.
6. Save and close the XML file.
7. To format the text displayed on the screen, modify the Templates file for the screen. To do this, refer to Formatting screen text.
8. Log on to Onyx Mobile and verify your changes.

Linking a primary contact to a company or incident record

In order to be able to link a primary contact to a company record, ensure that the following fields are present in the IndividualDetails.xml file:

- firstName
- lastName
- departmentDesc
- titleDescription
- emailAddress

If you do not want these fields to be displayed on the Details screen, you can set the attribute Visible=False for these fields.

Note: You cannot add or remove fields from the PrimaryContact group in the CompanyDetails.xml and the IncidentDetails.xml files.

Elements on a Details screen

Use the following table to configure properties for each group and field displayed on a Details screen.

Element	Attribute	Data Type	Description
EntityId	-	Numeric	Do not change this value. This node displays the number corresponding to the entity for which the details are displayed.
EntityName	-	String	Do not change this value. This node displays the name of the LBO that corresponds to the entity.
SubEntityName	-	String	Do not change this value. This node displays the name of the LBO that corresponds to the sub entity.
PrimaryId	-	-	Do not change this value. This is a placeholder for the primary ID of the LBO that corresponds to the Details screen. This value is retrieved when the Details screen is loaded.
PageTitle	-	String	Enter the text that will appear as the title for the Details screen.
EditPermission	-	-	
ResourceId	EditPermission	-	The UI Resource permission for editing the particular entity.
Value	EditPermission	-	By default, there should be no value. This contains the edit permissions of the particular entity.
Group	-	-	This element marks a section on the Details screen. Each group has a header and a detail section. You can add or remove groups on the Details screen. However, you cannot add multiple navigation menu groups to a single Details screen.
Group	Id	String	Enter a unique group ID for each group in the file. Text

Element	Attribute	Data Type	Description
			within the group is formatted based on the setting for this ID in the Template xml file. Do not change this value for the Do More group.
Group	Caption	String	Enter the text that will appear as the caption in the group header.
Group	IsCollection	Boolean	Enter True if the group is a collection object in OEAS. Enter False if the group is not a collection object in OEAS. If you have entered True, do not add fields to the group or move fields from such a group to a group where this value is False.
Group	IsPrimaryGroup	Boolean	Do not change this value.
Group	IsCount	Boolean	This attribute is available only for the Do More group. Enter True to enable users to view the number of records for each menu item in the group. Enter False to hide the record count for all menu items.
Row	-	-	This element marks a row on the Onyx Mobile screen. All fields within a single Row element appear in a single row on the Details screen.
Field	-	-	This element marks a field that is displayed on the Onyx Mobile screen.
Field	FieldName	String	Enter the name of the LBO column from which to retrieve data.
Field	DisplayName	String	Enter the caption for the field that will be displayed on the screen.
Field	Visible	Boolean	Enter True to enable users to view this field. Enter False to hide this field on the screen.
Field	Parent	-	If the field is a child field, then enter the LBO Column name of the parent field.
Field	MaskFormat	String	Enter a value to specify if the field validates a mask format for the value typed by the user.
Field	LboName	-	Enter the name of the LBO object to which the property belongs.
Field	PropertyPath	-	Enter the XML node path for the field within the LBO.
Field	Value	-	Do not edit this field.
Field	DataType	String	Enter the data type that is displayed on the screen for this field. The possible values are String, Int, GUIData, and DateTime.

Element	Attribute	Data Type	Description
Field	TapType	String	Enter a value for the action to be performed when a user taps this field. For a list of tap type values currently available, see Tap Type values. If a row contains multiple fields, then the TapType attribute can be enabled for only one of those fields.
Field	TapValue	-	Do not enter a value for this field. This value is retrieved when loading the screen.
Field	LinkColumnName	-	If the field can be tapped to display another Details screen, enter the FieldName value to which data in this field is linked. This field is not displayed on the screen.
Field	DomainData	-	Enter a value for the domain data. The possible values are referenceLookup, country, region, contactType, and user.
MenuItem	MenuItemId	Integer	Enter a unique value for each item in the navigation menu.
MenuItem	DisplayName	String	Enter the text that will appear as the caption for the menu item on the screen.
MenuItem	ResourceId	-	Enter the UI resource ID that is associated with the menu item in OES.
MenuItem	Action	String	Enter a value for the action to be performed when a user taps the menu item. For a list of action values currently available, see Action values.
MenuItem	PageTitle	String	Enter a value for the title of the screen that opens on tapping the menu item.
MenuItem	ContactTypeDid	Numeric	This attribute is used only for the Add to my favorites menu item. Enter the Contact Type ID value that corresponds to the contact type Favorite in OES Contact Type Administration under Internal\Company and Internal\Individual.
Filter	Property	-	Enter the name of the LBO column by which to filter data for the menu item. You can enter multiple properties to filter by.
Filter	Value	-	Enter the value that corresponds to the parameter that you want to filter the list by.
Filter	contextVariable	-	Enter a contextual value for the filter property, if it is dependent on the record displayed on the screen. This value is returned to OGS when a screen is loaded. Based on this value received, OGS sends the

Element	Attribute	Data Type	Description
			relevant data for the filter to Onyx Mobile. For a list of contextual values currently used, see Context Variable values.

Formatting Screen Text

You can use XML files to configure the formatting of the text displayed in each Onyx Mobile Detail and List screen. The default path for the template XML files is `C:\Program Files\Onyx\AppServer\Applications\Onyx\OnyxGatewayService\App_Data\Templates`.

Details screen template

Use the DetailsTemplate.xml file to specify font properties for the text displayed on each Onyx Mobile Details screen. The following table provides a list of the attributes that you can modify for each group on each screen. You can specify different values for each group. The values in the Default Value column are used if no values are specified by the administrator.

Attribute	Description	Data Type	Possible Values
Group Id	Enter a unique value for each group. This value is used to identify the group in a profile XML file that will display the text based on the formatting specified in for the group.	String	Header
CaptionFontWeight	The font style used to present the header caption text for the group. Enter Bold to change the font style to Bold, or leave blank to display as regular font.	String	Bold, Normal
CaptionFontSize	The font size used to present the header caption text for the group.	Integer	12
CaptionFontColor	The font color used to present the header caption text for the group.	Hexa decimal	#000000
CaptionBackground	The background color used in the header section for the group.	Hexa decimal	#000000
CaptionIsRoundedSection	The shape of the corners of the header section for the group.	Boolean	True, False
DataFontWeight	The font style used to present the data within the group.	String	Bold
DataFontSize	The font size used to present the data within the group.	Integer	12
DataFontColor	The font color used to present the data within the group.	Hexa decimal	#000000

Attribute	Description	Data Type	Possible Values
DataBackground	The background color used in the data section of the group.	Hexa decimal	#00000
DataIsRoundedSection	The shape of the corners of the data section of the group.	Boolean	True, False

List page template

Use the ListTemplate.xml file to specify font properties for the text displayed on each Onyx Mobile List screen. The following table provides a list of the attributes that you can modify for each group on each screen. You can specify different values for each group.

Attribute	Description	Data Type	Possible Values
Group Id	Enter a value in this field that is used in the profile XML file for Details, List, and Edit screens. The text within the group is formatted based on the values set here.	String	Title, Details
IsBold	The font style used to present the text within the group.	Boolean	True, False
FontSize	The font size used to present the text within the group.	Numeric	12

Add/Edit/Search page template

Use the MetadataTemplate.xml file to specify font properties for the text displayed on each of the Onyx Mobile Add/Edit/Search screens. The following table provides a list of the attributes that you can modify for each group on each screen. You can specify different values for each group. The values in the Default Value column are used if no values are specified by the administrator.

Attribute	Description	Data Type	Possible Values
Group Id	Enter a unique value for each group. This value is used to identify the group in a profile XML file that will display the text based on the formatting specified in for the group.	String	Header
CaptionFontWeight	The font style used to present the header caption text for the group. Enter Bold to change the font style to Bold, or leave blank to display as regular font.	String	Bold, Normal
CaptionFontSize	The font size used to present the header caption text for the group.	Integer	13
CaptionFontColor	The font color used to present the header caption text for the group.	Hexa decimal	#FFFFFF
CaptionBackground	The background color used in the header section for the	Hexa	#D00F3B

Attribute	Description	Data Type	Possible Values
	group.	decimal	
DataFontWeight	The font style used to present the data within the group.	String	Bold, Normal
DataFontSize	The font size used to present the data within the group.	Integer	12
DataFontColor	The font color used to present the data within the group.	Hexa decimal	#000000
ControlBackground	The background colors of the controls.	Hexa decimal	#FFFFFF

Understanding Tap Type Values

Tap Type values entered for a field in the XML file determine the action that is performed when a user taps the field on the screen. The following Tap Type values are currently implemented in Onyx Mobile.

Tap Type Value	Description
email	Launches the Compose Email screen of the default email client with the tapped email address in the To field.
phone	Dials the tapped phone number.
individual	Launches the Individual Details screen corresponding to the record tapped.
company	Launches the Company Details screen corresponding to the record tapped.
URL	Opens the site corresponding to the URL tapped within the Onyx Mobile app.

Understanding Action Values

Action values determine the action that is performed when a user taps a navigation menu item on the screen. The following Action values are currently implemented in Onyx Mobile.

Action Value	Description
internalContact	Launches the Internal Contacts screen, listing the internal contacts for the selected customer.
externalContact	Launches the External Contact screen, listing the external contacts for the selected customer.
AddMeAsFavorite	Adds the logged on user as a favorite internal contact for the selected customer, and adds the selected customer to the logged on user's list of favorite companies or individuals.

Action Value	Description
incident	Launches the Incident List screen listing the incidents based on the filters specified for the menu item.
map	Launches iPhone's map application displaying directions from the logged on user's current location to the primary address of the selected customer.
workNote	Launches an Incident List screen listing incidents that meet the filters specified for the navigation menu item.
task	Launches a Task List screen listing tasks that meet the filters specified for the navigation menu item.
addWorkNotes	Opens a screen where the user can add new work notes for the incident or task record.
addIncident	Opens a screen where the user can add a new incident for the customer record.

Understanding Context Variable Values

Context variable values define a context based on which records are filtered for a navigation menu list item. When a user taps a navigation menu item, these values are returned to OGS. Based on the value received, OGS sends the relevant data for the filter to Onyx Mobile.

The following context variable values are currently implemented in Onyx Mobile:

Context variable value	Description
$primaryId	Retrieves records that have the primary ID of the displayed record as the value of the property.
$Credential	Retrieves records based on the logged on user's Onyx session ID.
$appName	Retrieves records based on the OEAS application name.
$userId	Retrieves records that have the logged on user's ID as the value for the property.
$contentType	Retrieves records in the specified content format, for example JSON.
$siteId	Retrieves records that match the site ID specified in the app's configuration.
$lang	Retrieves records that match the language specified in the app's configuration.

8

Onyx Mobile Screens

Overview

Onyx Mobile includes the following screens to enable you to view and edit data in your Onyx CRM system. Click a link for more information about each screen.

Configuring Home Page

The Onyx Mobile Home Page contains a menu that enables you to navigate to different screens of the application.

Users can view lists of recent customers, favorite companies and individuals, add new customer records, and access pre-defined Web links. If you have configured user-specific mobile bookmarks, or if users have configured mobile bookmarks for their Navigator queries, these bookmarks are also displayed on the Home Page.

For information on configuring mobile bookmarks, see Configuring Mobile Bookmarks.

For information on configuring web links, see Configuring Web links.

Configuring Home Page

To modify how data is displayed on the Home Page, use the OnyxMobileHomePageMetadata.xml file. You can change the following attributes for each node.

Element	Attribute	Description
MenuItem	DisplayName	Enter the text to use as caption for the item.
MenuItem	Icon	Do not change this value. In the current implementation, your changes will not be reflected in the application.
MenuItem	Id	Do not change this value. In the current implementation, your changes will not be reflected in the application.
MenuItem	Name	Do not change this value. In the current implementation, your changes will not be reflected in the application.
MenuItem	SequenceIndex	Enter a numeric value to specify the sequence that the item will appear in on the menu.
MenuItem	URL	Do not change this value. In the current implementation, your changes will not be reflected in the application.

Element	Attribute	Description
MenuItem	Visible	Set to TRUE to display the item on the menu. Set to FALSE to hide the item.
-	IsCount	Set to TRUE to display the count of records for each mobile bookmark. Set to FALSE to hide the record count.
MenuItem	ResourceId	Contains UI Resource permissions. Enter the UI Resource value.

Configuring Recent Customers Screen

The Recent Customers screen displays a list of most recently visited customer records and cannot be further configured for Onyx Mobile. The number of records displayed is the same in Onyx for Desktop and Onyx Mobile, and depends on the value set in OES. For information on changing the number of records displayed, see the Onyx Technical Guide.

Configuring My Favorite Companies Screen

The Onyx Mobile Favorite Companies screen lists the company records for whom the logged-on user is listed as a favorite contact.

Configuring My Favorite Companies screen

To modify how data is displayed on the Favorite Companies page, use the FavoriteCompaniesMetaData.xml file. You can change the following attributes.

Element	Attribute	Description
sorting	isAscending	Set to TRUE to list favorite companies in ascending order. Set to FALSE to list companies in descending order.
sorting	numericSorting	Set to TRUE to sort based on numeric values. Set to FALSE to sort based on the text.
sorting	sortBy	Type the field name that you want to sort by. You can sort by: primaryid, secondaryid, objectType, suffix, and companyName.

Configuring My Favorite Individuals Screen

The Onyx Mobile Favorite Individuals screen lists the individual records for whom the logged-on user is listed as a favorite contact.

Configuring My Favorite Individuals screen

To modify how data is displayed on the Favorite Individuals page, use the avoriteIndividualsMetaData.xml file. You can change the following attributes.

Element	Attribute	Description
sorting	isAscending	Set to TRUE to list favorite individuals in ascending order. Set to FALSE to

Element	Attribute	Description
		list individuals in descending order.
sorting	numericSorting	Set to TRUE to sort based on numeric values. Set to FALSE to sort based on the text.
sorting	sortBy	Type the field name that you want to sort by. You can sort by: primaryid, secondaryid, objectType, suffix, salutation, firstName, middleName, and lastName.

Onyx Mobile List Screens

The Onyx Mobile List screens are read-only screens that display key information for a list of customer, incident, or task records. Tapping a record on a List screen opens the Details screen for the record where the user can view additional information and perform actions on the record.

Customer List screen

The Customer List screen displays a list of company and individual records that match specified search criteria. This screen appears when users run a search on Onyx Mobile, or when users tap a mobile bookmark on the Onyx Mobile Home Page.

External Contact List screen

The External Contact List screen displays a list of all companies and individuals that are external contacts for the record. The contact type is displayed below the contact name. This screen appears when users tap the View external contacts navigation menu item on the Company, Individual, or Incident Details screens.

Incident List screen

The Incident List screen displays a list of incidents that match specified criteria. This screen appears when users tap an Incident navigation menu item on the Company or Individual Details screen, or a mobile bookmark for incidents on the Home Page.

Internal Contact List screen

The Internal Contact List screen displays a list of internal contacts associated with a record. The contact type is displayed below the contact name. This screen appears when users tap the View internal contacts navigation menu item on the Company, Individual, or Incident Details screens.

Products List screen

The Products List screen displays a list of products that match specified criteria. This screen appears when users tap a Products navigation menu item on the Company or Individual Details screen, or a mobile bookmark for products on the Home Page.

Task List screen

The Task List screen displays a list of tasks that match specified criteria. This screen appears when users tap a Task navigation menu item on the Incident Details screen, or a mobile bookmark for tasks on the Home Page.

WorkNote List screen

The WorkNote List screen displays a list of work notes associated with an incident or a task. This screen appears when users tap the View worknotes navigation menu item on the Incident or Task Details screen.

The task explained in this topic is:

- Configure a List Screen

Adding/Editing Onyx Mobile Screens

The Onyx Mobile Add/Edit screens are where users can add or modify customer and incident records, add worknotes, or enter specific values to search for customer records.

Add or Edit Company Screen

The Add Company screen allows users to add a new company record to their Onyx CRM system using their iOS/Android device. Users can add details such as company name, address, phone, and email. Users can also link a primary contact to the company. The Edit Company screen allows users to modify previously entered information for a company record.

Add or Edit Individual Screen

The Add Individual screen allows users to add a new individual record to their Onyx CRM system using their iOS/Android device. Users can add details such as individual name, address, phone, and email. Users can also link a primary contact to the company. The Edit Individual screen allows users to modify previously entered information for an individual record.

Add or Edit Incident Screen

The Add Incident screen allows users to add a new incident record to their Onyx CRM system using their iOS/Android device. After selecting an incident category to create the incident, users can add details such as description, status, priority, type, recall date, and worknotes. The Edit Incident screen allows users to modify previously entered information for an incident record.

IncidenCategorList.xml contains the UI resource permission for adding an incident for a particular category. The xml contains the following:

- Resource ID - It contains the UI Resource permission value.
- Value - The resource permission of the user. Set always to nil.
- Key - The ID of the Incident Category.

Add Worknotes Screen

The Add Worknotes screen allows users to add a new worknote to an incident or task record in their Onyx CRM system using their iOS/Android device.

The task explained in this topic is:

- Configure an Edit Screen

Configuring Add/Edit Screens

The fields and actions available on each Add/Edit screen depend on the configuration within the XML file that corresponds to the screen. Use this information to configure XML files for Add/Edit screens.

Values you enter in the XML file are case-sensitive.

To configure Add/Edit screens:

1. Open the XML file corresponding to the Add/Edit screen that you plan to modify. For a list of XML files and the screens that they correspond to, see About XML files.
2. Configure attributes for each element within the XML file.
3. Configure attributes for each group within the XML file. You cannot add a new group to this screen.
4. Configure attributes for each field within each group.

Note: In the WorkNoteMetaData.xml file, do not remove the logMark and publishBitMark fields as these are essential to successfully save work notes.

5. Save and close the XML file.
6. To format the text displayed on the screen, modify the Templates file for the screen. To do this, refer to Formatting screen text.
7. Log on to Onyx Mobile and verify your changes.

Elements on an Add/Edit screen

Use the following table to configure properties for each group and field displayed on an Add/Edit screen.

Element	Attribute	Data Type	Description
EntityId	-	Numeric	Do not change this value. This node displays the number corresponding to the entity for which the list is displayed.
EntityName	-	String	Do not change this value. This node displays the name of the LBO that corresponds to the entity.
PrimaryId	-		Placeholder for the primary ID of the LBO that corresponds to the Edit screen. This value is retrieved when the Edit screen is loaded. Do not edit this value.
AddPageTitle	-	String	Enter the text that will appear as the title for the screen when users open it to add a new record.
EditPageTitle	-	String	Enter the text that will appear as the title for the

Element	Attribute	Data Type	Description
			screen when users open it to edit an existing record.
EditPermission	-	-	
ResourceId	EditPermission	-	The UI Resource permission for editing the particular entity.
Value	EditPermission	-	By default, there should be no value. This contains the edit permissions of the particular entity.
Group	-		This element marks a section on the Edit screen. Each group has a header and a detail section.
Group	Id	String	Enter a unique group ID for each group in the file. Text within the group is formatted based on the setting for this ID in the Template xml file.
Group	Caption	String	Enter the text that will appear as the caption in the group header.
Group	IsCollection	Boolean	Enter True if the group is a collection object in OEAS. Enter False if the group is not a collection object in OEAS. You must add fields for a collection object group that users can update. If you have entered True, do not add fields to the group or move fields from such a group to a group where this value is False.
Group	IsPrimaryGroup	Boolean	Do not change this value.
Field	-	-	This element marks a field that is displayed on the Onyx Mobile screen.
Field	FieldName	String	Enter the name of the LBO column from which to retrieve data.
Field	DisplayName	String	Enter the caption for the field that will be displayed on the screen. Do not enter a value here if the default value for this attribute is blank.
Field	Visible	Boolean	Enter True to enable users to view this field. Enter False to hide this field on the screen. Do not enter a value here if the default value for this attribute is blank.
Field	Type	String	Enter the data type that is displayed on the screen for this field.
Field	Parent	String	If the field is a child field, then enter the LBO Column name of the parent field. For example, Country is a parent field for State.

Element	Attribute	Data Type	Description
Field	ControlType	String	Enter the control type to use for the field. The possible values are TextBox, ComboBox, DropDown, CheckBox, URL, Link, RadioButton.
Field	DisableLinkColumn	String	This attribute is used only for the Primary Contact firstname field. Enter the dependent fieldname that will become uneditable when this field is populated by linking it to another record. For example, when a company is linked to a primary contact, the primary contact's last name field is populated automatically based on the linked contact's details. At this time, users should not be able to type in a different last name. The fieldname you enter here should also be listed in the DependentFieldName attribute for the field.
Field	MaskFormat	String	Enter a value to specify if the field validates a mask format for the value typed by the user. Currently this attribute is used for the postal code and phone fields.
Field	IsRequired	Boolean	Enter TRUE if a value is needed in the field to save the record. Enter FALSE if the field can be left empty.
Field	IsEditable	Boolean	Enter TRUE to enable users to enter values in the field. Enter FALSE to make the field read-only.
Field	CacheName	String	Enter the name of the OEAS Reference Lookup table.
Field	CacheParentId	-	Do not change this value. In the current implementation, your changes will not be reflected in the application.
Field	LboObjectName	String	Enter the name of the LBO object to which the property belongs.
Field	PropertyPath	String	Enter the XML node path for the field within the LBO.
Field	LboProperty	String	Enter the name of the LBO object to which the property belongs.
Field	linkPropertyPath	String	Enter the name of the LBO property for the primary contact ID in the CompanyMetaData.xml file, or the parent company ID in the IndividualMetaData.xml file.
Field	linkEntity	Integer	Enter '1' to indicate a link to an Individual record. Enter '2' to indicate a link to a Company record.

Element	Attribute	Data Type	Description
Field	TypeIdProperty	String	Enter the LBO property name for the primary value selected for a collection field.
Field	TypeIdValue	Integer	Enter the ID that corresponds to the collection field's lookup value. To get this value, in OES Reference Table Administration, click the Lookup Value field, and copy the value from the ID of Lookup Value field. This attribute is mandatory to ensure that collection fields display accurate values.
Field	LinkFieldName	String	Enter the LBO field name for the primary contact or parent company.
Field	DependentFiedName	String	Enter the field name for the primary contact's details from the Company LBO and associate it with the field name for the Individual's details from the Individual LBO, in the format contactFirstName;firstName. You must enter this data for the following fields: First Name, Last Name, Department, Title, and Email. All dependent fields should also be present in IndividualDetail.xml.
Field	OnyxTimeStamp	-	Do not change this value. This value is dynamically populated with the user's Onyx session ID when a call is made to OGS.
Field	MaxLength	Integer	Enter the maximum field length.
Field	Value	-	Do not enter a value for this field. The value is retrieved when loading the screen.
Strings	-	-	This element is not used in the current implementation.

Configuring Customer Search Screen

The Customer Search screen allows users to enter specific criteria to search for company and individual records from their iOS or Android device. Users can perform a search based on several criteria such as ID, name, address, email, location, and type.

The fields and actions available on the Search screen depend on the configuration within the XML file that corresponds to the screen. Use this information to configure the XML file for the Customer Search screen. To configure the fields that are displayed in the Customer Search Result List screen, see Configuring List screens.

Configuring Customer Search screen

To modify how data is displayed on the Customer Search screen, use the CustomerSearchMetaData.xml file. Use the following table to configure properties for each search criteria field. You cannot add a new group to this screen.

Element	Attribute	Description
Group	Id	Contains the value for the group ID. Do not edit this value.
Group	Caption	Enter the text that will appear as the group header caption. The default caption is Search Criteria.
Group	lboName	Contains the name of the LBO that is used to validate search criteria. Do not edit this value.
Group	methodName	Contains the name of the method that is used to retrieve data based on the search criteria entered. Do not edit this value.
Field	FieldName	Enter the field name that you want to use as a search criterion.
Field	DisplayName	Enter the caption for the field that will be displayed on the screen.
Field	Value	
Field	Type	Enter the data type for the field. The possible values are CHAR255, BOOLEAN, INTEGER.
Field	Parent	If the field is a child field, then enter the FieldName value of the parent field.
Field	SequenceIndex	Enter a numeric value to specify the sequence that the item will appear on the screen.
Field	ControlType	Enter the control type to use for the field. The possible values are TextBox, ComboBox, DropDown, CheckBox, URL, Link, RadioButton.
Field	MaskFormat	Enter a value to specify if the field validates a mask format for the value typed by the user. The possible values are PostalCode, Email, Phone.
Field	IsRequired	Enter 'True' if a value must be entered in this field to run a successful search. Enter 'False' if entering a value in the field is optional.
Field	IsEditable	Enter 'True' to enable users to enter values in the field. Enter 'False' to make the field read-only.
Field	Visible	Enter 'True' to enable users to view the field on the screen. Enter 'False' to hide the field.
Field	CacheName	Enter the name of the OEAS Reference Lookup table for reference fields if applicable.
Field	CacheParentId	Do not change this value. In the current implementation, your changes will not be reflected in the application.
Field	PropertyPath	Enter the XML node path for the field in the LBO.

Element	Attribute	Description
Field	LBOObjectName	Enter the name of the LBO that the field is associated with.
Field	LboProperty	Enter the name of the LBO property that is mapped to the FieldName.
Field	TypeIdProperty	Enter the LBO property name for the primary value selected for a collection field.
Field	TypeIdValue	Enter the type ID value to match with OEAS. This attribute is needed only for collection fields.
Field	maxLength	Enter the maximum field length.
Strings		Do not enter any attributes for this node.

Understanding XML Files for Detail Screens

The fields and actions available on each List screen depend on the configuration within the XML file that corresponds to the screen. Use this information to configure XML files for List screens.

Note: Values you enter in the XML file are case-sensitive.

To configure List screens:

1. Open the XML file corresponding to the List screen that you plan to modify. For a list of XML files and the screens that they correspond to, see About XML files.
2. Configure attributes for each element within the XML file.
3. Configure attributes for each group within the XML file. You can add or remove groups on this screen.
4. Configure attributes for each field within each group.
5. Save and close the XML file.
6. To format the text displayed on the screen, modify the Templates file for the screen. To do this, refer to Formatting screen text.
7. Log on to Onyx Mobile and verify your changes.

Use the following table to configure properties for each group and field displayed on a List screen.

Element	Attribute	Data Type	Description
EntityId	-	Numeric	Do not change this value. This node displays the number corresponding to the entity for which the list is displayed.
EntityName	-	String	Do not change this value. This node displays the name of the LBO that corresponds to the entity.
LBOName	-	-	Do not change this value. This node displays the

Element	Attribute	Data Type	Description
			name of the LBO from which data is retrieved for the list.
IsBookmarkList	-	Boolean	
PageTitle	-	-	Do not change this value. This is a placeholder for the name of the navigator query that is marked as a mobile bookmark.
PageSize	-	-	Enter the number of records that will be retrieved and displayed on the screen at a time. When users scroll down the list, subsequent calls are made to retrieve the remaining records.
PageCount	-	-	Do not enter a value here. This node is a placeholder for the number of calls to be made to retrieve records. This value changes dynamically when a menu item is tapped, based on the screen size of the device, the page count, and the total number of records to be retrieved.
CurrentPage	-	-	Do not change this value. This node is a placeholder for the screen displayed on the device.
PageCallId	-	-	Do not change this value. This node displays the ID assigned to the screen in the database.
LBOMethodName	-	-	Displays the name of the LBO method that is used to retrieve data for the list. You can enter the name of the method that you want to use.
AddPermission			
ResourceId	AddPermission	-	The UI Resource permission for adding the particular entity.
Value	AddPermission	-	By default, there should be no value. This contains the add permissions of the particular entity.
CategoryID	AddPermission	int	Do not change this value. This node is used to find the category of the list screen at runtime.
Group	-	-	This element marks a section on the List screen. The information displayed on each List screen is divided into two groups. Each group is identified by a node in the XML file. You can add or remove groups for a List screen, and specify the data that is displayed for each group.
Group	Id	-	Enter a unique group ID for each group in the file. Text within the group is formatted based on the

Element	Attribute	Data Type	Description
			setting for this ID in the Template.xml file.
Field	-	-	This element marks a field that is displayed on the Onyx Mobile screen.
Field	FieldName	String	Enter the name of the LBO column from which to retrieve data.
Field	DisplayName	String	Enter the caption for the field that will be displayed on the screen.
Field	Visible	Boolean	Enter True to enable users to view this field. Enter False to hide this field on the screen.
Field	Parent	-	If the field is a child field, then enter the LBO Column name of the parent field.
Field	MaskFormat	String	Enter a value to specify if the field validates a mask format for the value typed by the user.
Field	LboName	-	Enter the name of the LBO object to which the property belongs.
Field	PropertyPath	-	Enter the XML node path for the field within the LBO.
Field	DomainData	-	Enter a value for the domain data. The possible values are referenceLookup, country, region, contactType, and user.
Field	DataType	String	Enter the data type that is displayed on the screen for this field.
Field	ViewColumnName	-	This attribute is used for mobile bookmark list screens. Enter the value used for field_name corresponding to the field in the corresponding entity view. For more information about the value in this field, see Configuring list screens for mobile bookmarks.

Configuring XML File for Web Links

The Web Links page contains links to specific Websites from within Onyx Mobile. Tapping a Web link opens the referenced Website within Onyx Mobile. Use this information to configure the URI details for the Websites that you want users to access from Onyx Mobile.

To configure Web links:

1. Open the WebLinks.xml file. For a list of XML files and the screens that they correspond to, see About XML files.
2. Configure attributes for the weblink element within the XML file. Add a row for each additional link that you want to create.
3. Save and close the XML file.
4. Log on to Onyx Mobile and verify your changes.

Elements on the Web Links screen

Use the following table to configure properties for each group and field displayed on the Web Links screen.

Element	Attribute	Description
weblink	DisplayName	Enter user-friendly text to display on the Web Links screen for the URI. For example, Aptean.
weblink	TargetURL	Enter the value of the URL to launch when the link is tapped.

Configuring Security

Using OES Security Administration, you can secure access to Onyx Mobile and its various features. By granting and denying access to UI resources, you control the actions that your users can perform.

Set access permissions to UI resources in order to:

- Grant or deny access to Onyx Mobile
- Grant or deny access to navigation menu items on Details screens

You can use the default UI resources included in your Onyx CRM system, or you can create new resources based on your organization's business requirements. By default, all users are denied permission to the new UI resources created for Onyx Mobile. You must grant access based on your users' needs. It is a good practice to secure UI resources based on roles, and then set more restrictive permissions for users within each role.

Resource used for Onyx Mobile logon

Resource Name	Function
UI.Mobile.logon	This resource provides access to Onyx Mobile. Users without permission to this resource cannot log on to Onyx Mobile.

Resources used for Onyx Mobile

Resource Name	Function
UI:OM.menu.product	This resource provides access to menu items that open Product list screens. Users without permission to this resource cannot view menu items for Product lists on Details screens.
UI:OM.menu.task	This resource provides access to menu items that open Task list screens. Users without permission to this resource cannot view menu items for Task lists on Details screens.
UI:OM.menu.incident.sales	This resource provides access to menu items that open Sales Incident list screens. Users without permission to this resource cannot view menu items for Sales Incident lists on Details screens.
UI:OM.menu.incident.service	This resource provides access to menu items that open Service Incident list screens. Users without permission to this resource cannot view menu items for Service Incident lists on Details screens.
UI:OM.menu.incident.support	This resource provides access to menu items that open Support Incident list screens. Users without permission to this resource cannot view menu items for Support Incident lists on Details screens.
UI:OM.menu.incident	This resource provides access to menu items that open Incident list screens. Users without permission to this resource cannot view menu items for Incident lists on Details screens.
UI:OM.company.edit	This resource provides access to edit the company record. Users without permission to this resource cannot edit the company record.
UI:OM.company.add	This resource provides access to add the company record. Users without permission to this resource cannot add the company record.
UI:OM.individual.edit	This resource provides access to edit an individual record. Users without permission to this resource cannot edit an individual record.
UI:OM.individual.add	This resource provides access to add an individual record. Users without permission to this resource cannot add an individual record.
UI:OM.incident.add	This resource provides access to choose a category to add an incident. Users without permission to this resource cannot choose the category.

Resource Name	Function
UI:OM.incident.sales.edit	This resource provides access to edit a sales incident. Users without permission to this resource cannot edit a sales incident.
UI:OM.incident.service.edit	This resource provides access to edit a service incident. Users without permission to this resource cannot service a sales incident.
UI:OM.incident.support.edit	This resource provides access to edit a sales incident. Users without permission to this resource cannot edit a support incident.
UI:OM.incident.support.add	This resource provides access to add a sales incident. Users without permission to this resource cannot add a support incident.
UI:OM.incident.service.add	This resource provides access to add a service incident. Users without permission to this resource cannot add a service incident.
UI:OM.incident.sales.add	This resource provides access to add a sales incident. Users without permission to this resource cannot add a sales incident.
UI:OM.menu.incident.sales.worknote.add	This resource provides access to menu items that to add a worknote for Sales Incidents. Users without permission to this resource cannot add worknote for a sales incident.
UI:OM.menu.incident.service.worknote.add	This resource provides access to menu items that to add a worknote for Service Incidents. Users without permission to this resource cannot add worknote for a service incident.
UI:OM.menu.incident.support.worknote.add	This resource provides access to menu items that to add a worknote for Support Incidents. Users without permission to this resource cannot add worknote for a support incident.
UI:OM.menu.task.service.worknote.add	This resource provides access to menu items that to add a worknote for Service Tasks. Users without permission to this resource cannot add worknote for a service task.
UI:OM.menu.task.sales.worknote.add	This resource provides access to menu items that to add a worknote for Sales Tasks. Users without permission to this resource cannot add worknote for a sales task.
UI:OM.menu.task.support.worknote.add	This resource provides access to menu items that to add a worknote for Support Tasks. Users without permission to this resource cannot add worknote for a support task.
UI:OM.externalcontact.add	This resource provides access to add an external contact. Users without permission to this resource cannot add an external contact.

Resource Name	Function
UI:OM.internalcontact.add	This resource provides access to add an internal contact. Users without permission to this resource cannot add an internal contact.
UI:OM.menu.sales.task	This resource provides access to menu items that open Sales task lists screens. Users without permission to this resource cannot view menu items for Sales task lists on Details screens.
UI:OM.menu.service.task	This resource provides access to menu items that open Service task lists screens. Users without permission to this resource cannot view menu items for Service task lists on Details screens.
UI:OM.menu.support.task	This resource provides access to menu items that open Support task lists screens. Users without permission to this resource cannot view menu items for Support task lists on Details screens.
UI:OM.task.sales.edit	This resource provides access to edit a sales task. Users without permission to this resource cannot edit a sales task.
UI:OM.task.service.edit	This resource provides access to edit a service task. Users without permission to this resource cannot edit a service task.
UI:OM.task.support.edit	This resource provides access to edit a support task. Users without permission to this resource cannot edit a support task.
UI:OM.task.sales.add	This resource provides access to add a sales task. Users without permission to this resource cannot add a sales task.
UI:OM.task.service.add	This resource provides access to add a service task. Users without permission to this resource cannot add a service task.
UI:OM.task.support.add	This resource provides access to add a support task. Users without permission to this resource cannot add a support task.

Creating and Registering Class Assembly

In SQL Adapter, use the sample project 'oaCustomCLRAssembly' to create a new DLL for your custom SQL CLR trigger. This project is included in your installation package under Customization Support>Database Server>Custom_CLR_Trigger.

The sample project contains code to create a notification for the event: New product is created in Onyx. Use this as a reference to create your own custom DLL.

Prerequisites

Before creating the class assembly, be sure you have the following information for each custom notification:

- NotificationId: Value of the secondaryId column of the NotificationType table corresponding to the custom notification.
- Onyx site ID
- Trigger Name for the notification

Creating the class files and DLL

To create class files for the new notification:

1. Open the caCustomCLRAssembly sample project included in your installation package.

   
   Note: Do not remove the “oaServiceLibrary” DLL reference from the sample project.

2. Create a new class file within the project and give it an appropriate name.
3. In the newly created class file, create a public partial class named OnyxCLRTriggers for your new notification.
4. Modify the attribute declaration for the CLR trigger with the parameters required for your notification.
   - **Name**. Enter the name of the SQL CLR Trigger created on the database table.
   - **Target**. Enter the name of the database table in OEDB on which you to create the notification.
   - **Event**. Enter the action on the record that will initiate the event notification.
5. Write a function to implement the logic for the new notification as implemented in the sample project. In the NotificationId variable, enter the value from the secondaryId column of the

NotificationType table corresponding to the custom notification.

6. Repeat steps 2 to 5 for each custom notification that you want to create.
7. Build the project and then build the solution. The compiled .DLL file is saved in the Bin\Release folder of the sample project.
8. Copy the newly created DLL and paste it within your Onyx database installation folder. The default path for this is `C:\Program Files(x86)\Onyx\Notification_CLR.`

Registering the assembly

Run the following script to register the assembly on your OTDB database, replacing the text in brackets with the appropriate values.

---/// Register the Onyx Assembly in SQL Server

Sample Script

```
CREATE ASSEMBLY [Name of the Assembly to be registered in SQL Server, starting with ca, to indicate custom trigger]
AUTHORIZATION [DBAdministrator schema]
From ['DLL File Name with Path']
WITH PERMISSION_SET = UNSAFE
GO
```

Example script to register the assembly

```
CREATE ASSEMBLY [caProductNotification]
AUTHORIZATION [dbo]
From 'C:\Program Files\Onyx\Notification_CLR\caProductNotification.dll'
WITH PERMISSION_SET = UNSAFE
GO
```

Creating SQL CLR Trigger

After registering the assembly, create SQL CLR triggers for each custom notification. To do this, use the following sample script, replacing the text in brackets with the appropriate values.

```
CREATE TRIGGER [Name of trigger]
ON [Database Table Name on which the trigger is created]
FOR [Actions on which trigger should fire]
AS EXTERNAL NAME [DLL Name].[Class Name ].[Function Name]
GO
Example Script :
---///New Product created- 1
```

```
CREATE TRIGGER ctProductNotification
ON incident
FOR INSERT, UPDATE
AS EXTERNAL NAME
cpProductNotificationcaProductNotification.OnyxCLRTriggers.ProductCreat
edInOnyxPublic
GO
```

Entering Resource String Values

Create one entry for each custom search type in the Telerik.Resources project to specify the label to identify the custom entity in tool tips and messages.

To enter resource string values:

1. In your development environment, ensure that you have installed the following programs on your computer:
   - The latest version of Microsoft Visual Studio 2012 with the latest updates available from Microsoft
   - Microsoft Silverlight 5.0
   - RadControls for Silverlight Q3 2013
2. In your Onyx 7.8 installation package, go to Customization Support\Web Server\.

   Note: To modify resource strings for a specific language, use the installation package for the corresponding language.
3. From the **Web Server** folder, copy the **Telerik.Resources** project to the desired folder on your development computer.
4. From the **Telerik.Resources** project, open the solution file **Telerik.Resources.sln** in Visual Studio 2012.
5. In Solution Explorer, open the **ControlResource.resx** file.
6. Modify an entry for each value for the corresponding language.
   - Resource String Name. Ensure the value is not edited.
   - Value = <desired display value in the installation language>
7. Set the configuration mode to Release. To do this, select **Build>ConfigurationManager option>Active Solution Configuration**
8. Build the solution.

9. In the **Telerik.Resources** project folder, navigate to **\Telerik.Resources\Bin\Release**, and copy the updated **Telerik.Resources.dll**.
10. Paste the Telerik.Resources.dll into the **ClientBin** folder within the OEP installation folder. The default location for the folder is `C:\Program Files\Onyx\EmployeePortal\QuickSearch\ClientBin`.
11. Clear all cache and temporary Internet files on each client computer.

Note: Be sure to back up the modified project. You will need it when you want to make further modifications and when you want to upgrade to a newer version of Onyx.

Limitations

The following are current system limitations while using Onyx.

Result grid is blank when the maximum length of Navigator Search results exceeds supported limit

If the length of the Navigator search results string exceeds 3800 characters, no records are returned and the Result Grid is blank. This happens due to a technical limitation.

Troubleshooting

This section includes information on the common errors that may occur when installing the core components of your Onyx system, and how you can resolve those errors.

OPS Shared Printer

When an active shared printer is configured on an OPS server with Microsoft Word 2010 installed, the Print subsystem throws errors.

For information on correctly configuring the printer, see the *Onyx Installation Guide*.

Onyx menu may not appear in the appointment form

The Onyx menu may not appear in the appointment form accessed from Microsoft Outlook Calendar for an appointment. This situation occurs for appointments created directly on the calendar by entering text within a selected block of time.

To display the Onyx menu in this situation, save, close, and then reopen the appointment. To prevent this situation, create appointments using the appointment form (displayed by using the File menu or by double-clicking the desired block of time on the calendar).

Restoring deleted appointments in Microsoft Outlook does not work as expected in OEP

Using the Undo feature in Outlook to restore a deleted appointment does not update the appointment in OEP.

To implement this change, the user must open the appointment through Outlook and send an update. After the appointment is saved in OEDB, the appointment updates in OEP.

Launching Navigator on a different domain

To launch Navigator on a different domain than the OEP installation, provide the machine name instead of the IP address when accessing the OEP URL.

Error loading Navigator filters on the Home page

If no search parameters are stored in the database for a Navigator filter that is saved to Home page, an error occurs when loading the Navigator Filters section on the Home page.

To resolve this issue, run the following statement on the Onyx Persistence database:

select query_name from query where query_parameter is null or query_parameter=''

This query returns the Navigator filters that have all blank parameters. Delete these queries from Navigator.

Regional Settings

If a user who is logged on to Onyx changes the date format under the Region and Language Settings of the client computer, the changed format will not reflect within Onyx until the user has logged out and then logged back on to Onyx.

Message appears when attempting to perform an action after loading a filter that has date as a search criterion

After loading a filter that contains date as a criterion, if you try to perform any further action, a message appears asking if you would like to save changes made to the filter. Click OK on the message and continue working with Navigator.

Email send failures with local pickup option of local SMTP service

If you use the local pickup option of the local SMTP service for sending email messages via the OEAS or OPS server, you may see the following error message: "Failed to send the Email using Collaboration Data Objects (CDO). HR: 80040222; Description: The pickup directory path is required and was not specified."

For details on correcting this, see Microsoft Knowledge Base article 816789.

OPS cannot send Lotus Notes email messages over the Internet to an EMS mail-box

Lotus Notes includes a delivery preference that allows a Notes user to send email messages to other Notes users over the Internet. Monitored EMS mailboxes cannot process messages that are sent with this delivery preference. If you get an error that says that an invalid character was found in the text content of an email message, this issue might be the cause of the error.

OPS timeouts send HasPermission email

After a timeout and successful restart, OPS sends an email notification such as "The HasPermission method on the OnyxSessionManager component failed for the "Onyx" application, with the following error message [message text]." These informational messages explain the reason for the OPS timeout and restart.

Incident and Task category description in Summary and Individual email

Currently the data for Incident and Task category description is taken from reference_parameter_ml table. If the description needs to be modified the data in the table has to be modified. If data is to be taken from other tables like Incident_category_ml or Task_category_ml modifications has to be done on the EMF process and OGS extension. For more information on customizing EMF refer to the Onyx 7.5 Technical guide.

If a role is removed from 'UI:OEP:Notifications' user continues to get a mail when the event occurs

To resolve this issue:

1. In the configuration section of the SQL script give the context site id of the web server

SET @context_site_id=1

2. Run the Notification Fix.sql which is available in the folder path \Customization Support\Database Server\Notification Fix.

Internet Explorer 10 compatibility mode setting

To make Onyx OEP compatible with Internet Explorer 10:

1. Open Internet Explorer 10 browser.
2. Go to **Tools**, click on **Compatibility View Settings**.
3. In the **Compatibility View Settings** dialog, enter your Onyx OEP website details.

9

Scripting Extensibility

Overview

A script is an end-user interface with which users can:

- Guide their interactions with customers.
- Gather, create, update, or disseminate data across OEAS.
- Call other OEAS features to perform tasks, including those that otherwise require opening and working with OES Administration Workbench tools.

Abbreviations

The following abbreviations are used in this chapter:

- SS: Script Step
- SP: Step Prompt
- OEAS: Onyx Enterprise Application Server
- OES: Onyx Enterprise Studio
- OGS: Onyx Gateway Service
- SDK: Software Development Kit

Client Configuration

Following are the files that are used to do client configuration, and are located at `C:\ProgramFiles\Onyx\EmployeePortal\scripting_ngen\configuration`:

- WebApiConfiguration.js
- ScriptingWebUIConfiguration.js
- ScriptingWebApiConstants.js
- ScriptingMessageConstants.js
- ScriptingCustomization.js
- SurveyWebApiConstants.js
- ScriptingController.js

WebApiConfiguration.js

- BASE_URI - Web API base URI configuration for scripting application.
  - Base URI for OEP scripting (integrated) is " ../ "

    Example: http:// WS003LT0094PRD/OEP_Onyx
  - Base URI for Scripting stand alone is empty string ("") that is installed directory.
- REQUEST_TIMEOUT - API request time out.

ScriptingWebUIConfiguration.js

This file is for any UI configuration that users prefer to change. Currently there is only one configuration added to this file for string casing. This builds over time.

Example: Contact name may vary all uppercase or all lowercase based on UI_UPPERCASE property set to true or false.

ScriptingWebApiConstants.js

All the API calls to the server for Onyx Scripting is present here as constant. These API calls are set to their default value. If the user wants to make some server side changes, the customized API should be pointed to `Onyx.UI.Scripting.Controller.dll` file. For more details about scripting related controllers in the scripting server controller, click Scripting Server Controller Extensibility.

ScriptingMessageConstants.js

In case of adding a Sencha extension to the browser, LogStore is created to track any error encountered.

The messaging string constants are present in the following image:

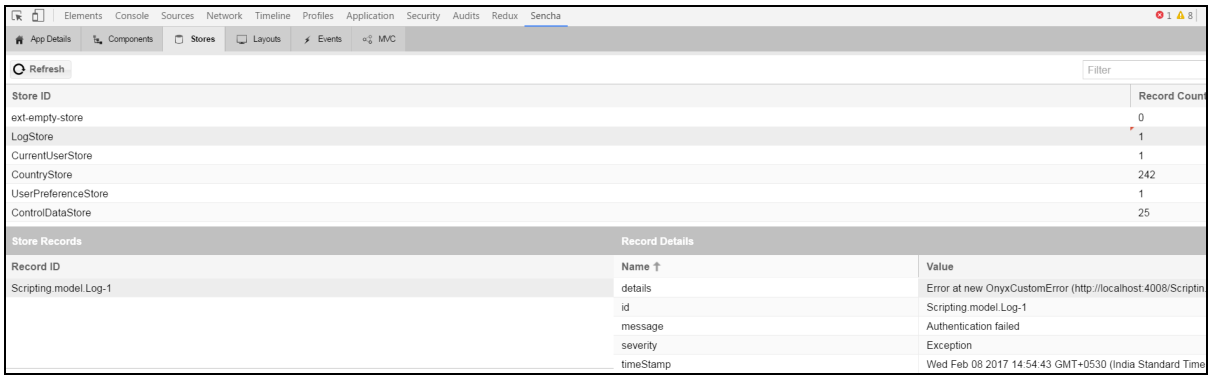


Figure 9-1: LogStore

ScriptingCustomization.js

This file holds the name mapping to the controller that is pulled in dynamically to the application.

The default setup is `CONTROLLER_NAME: 'ApplicationConfiguration.ScriptingController'`. `ScriptingController` is the file where the custom code goes.

SurveyWebApiConstants.js

All the API calls to the server for Onyx Scripting - Survey Control is present here as constant. These API calls are set to their default value.If the user wants to make some server side changes, the customized API should be pointed to `Onyx.UI.Survey.Controller.dll` file. For more details about scripting related controllers in the scripting server controller, click Scripting Server Controller Extensibility.

ScriptingController.js

The `ApplicationConfiguration.ScriptingController` file is defined here. There are three methods provided wherein you can inject logic to change the behavior. Following are the three methods that you can use for customization :

1. **InitializeScript**

InitializeScript method is used when there is any application level changes required. This method is called after the user is successfully authenticated. `this.callParent()` should be the last line of code for this method.

2. **AfterRender**

   AfterRender method is used to customize all the controls that do not use callback or on demand load (Lazying Loading of data). `this.callParent()` should be the last line of code for this method.For more details about On Application Load Controls, click On Application Load Prompts.

3. **AfterLazyLoading**

   AfterLazyLoading method is used to customize all the controls that use callback or on demand load (Lazying Loading of data). For more details about On Lazy Load Controls, click On Lazy Loading Prompts.

Example

In the following example, a script is created based on customer type selection in the first step and credit card selection value to be filtered in the second step.

In the first step we have an option to select based on the Script Designer configuration as follows:

- Customer
- Prospect
- Contact

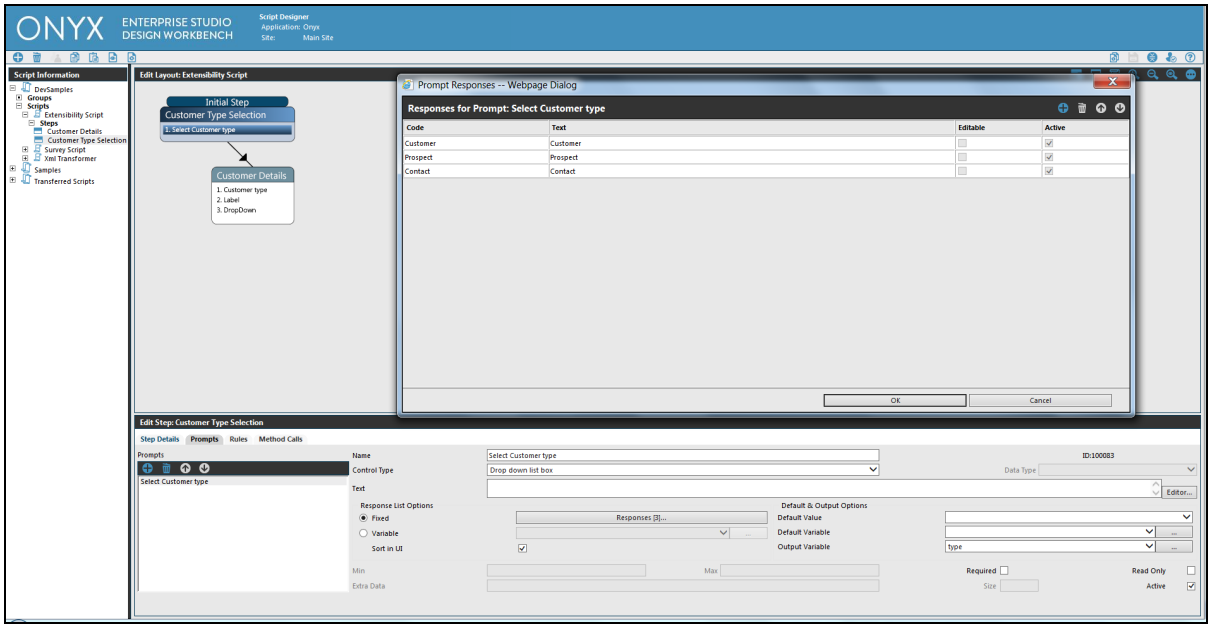


Figure 9-2: Responses for Prompt - Select Customer type

The second step has the following configurations for credit card type:

- Visa
- Master
- Mastro
- Rupay
- American Express

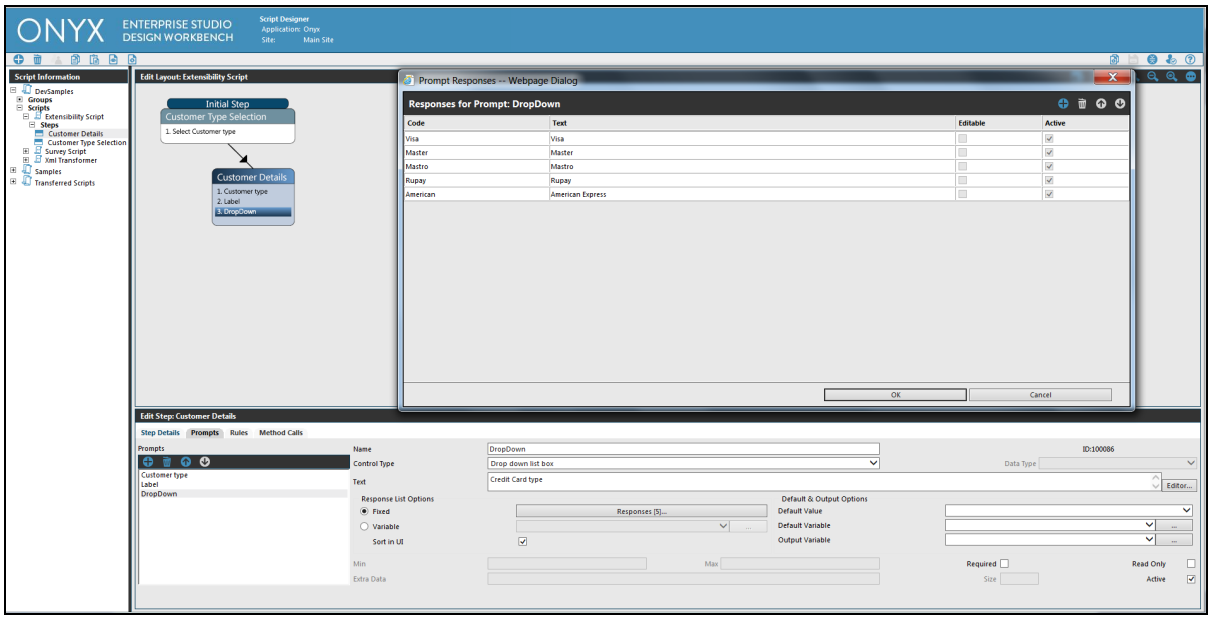


Figure 9-3: Responses for Prompt - DropDown

Inject some custom code to filter the dropdown values for the credit type based on the selection of customer type on the first step.

Since the dropdown control is lazy loaded control using Ext.store, we have written the code in AfterLazy.

```
AfterRender: function AfterRender() {
    this.callParent();
},
AfterLazyLoading: function AfterLazyLoading() {
    var scriptActiveStepId = Ext.ComponentQuery.query('#' + Scripting.internal.ScriptIdentifierConstants.SCRIPT_CENTER_CONTAINER)[0].getActiveTab().getItemId().split('_')[1];
    if (scriptActiveStepId == '100030') {
        var cInfo = Ext.ComponentQuery.query('#' + Scripting.internal.ScriptIdentifierConstants.STEP_ACTION_PANEL_PREFIX +
          scriptActiveStepId)[0].items.items[0].items.items[1].getValue();

        var creditCardControl = Ext.ComponentQuery.query('#SP_100086');

        if (cInfo == "Prospect") {
            var record = creditCardControl[0].store.getData().items.find(Scripting.internal.ScriptingHelper.multiPropertyCompare,
             { compareWith: 'Text', value: "Rupay" });
            if (record)
                creditCardControl[0].store.remove(record);
        }
        else if (cInfo == "Customer") {
            var record = creditCardControl[0].store.getData().items.find(Scripting.internal.ScriptingHelper.multiPropertyCompare,
             { compareWith: 'Text', value: "Visa" });
            if (record)
                creditCardControl[0].store.remove(record);
            var record = creditCardControl[0].store.getData().items.find(Scripting.internal.ScriptingHelper.multiPropertyCompare,
            { compareWith: 'Text', value: "Master" });
            if (record)
                creditCardControl[0].store.remove(record);
        }
        else if (cInfo == "Contact") {
            var record = creditCardControl[0].store.getData().items.find(Scripting.internal.ScriptingHelper.multiPropertyCompare,
             { compareWith: 'Text', value: "Mastro" });
            if (record)
                creditCardControl[0].store.remove(record);
            var record = creditCardControl[0].store.getData().items.find(Scripting.internal.ScriptingHelper.multiPropertyCompare,
            { compareWith: 'Text', value: "Master" });
            if (record)
                creditCardControl[0].store.remove(record);
            var record = creditCardControl[0].store.getData().items.find(Scripting.internal.ScriptingHelper.multiPropertyCompare,
           { compareWith: 'Text', value: "American Express" });
            if (record)
                creditCardControl[0].store.remove(record);
            var record = creditCardControl[0].store.getData().items.find(Scripting.internal.ScriptingHelper.multiPropertyCompare,
          { compareWith: 'Text', value: "Visa" });
            if (record)
                creditCardControl[0].store.remove(record);
        }
```

Figure 9-4: Custome Code Example

Step 1: To verify the changes, log on to Scripting and select **Customer** from ACTION panel as shown in the following image.

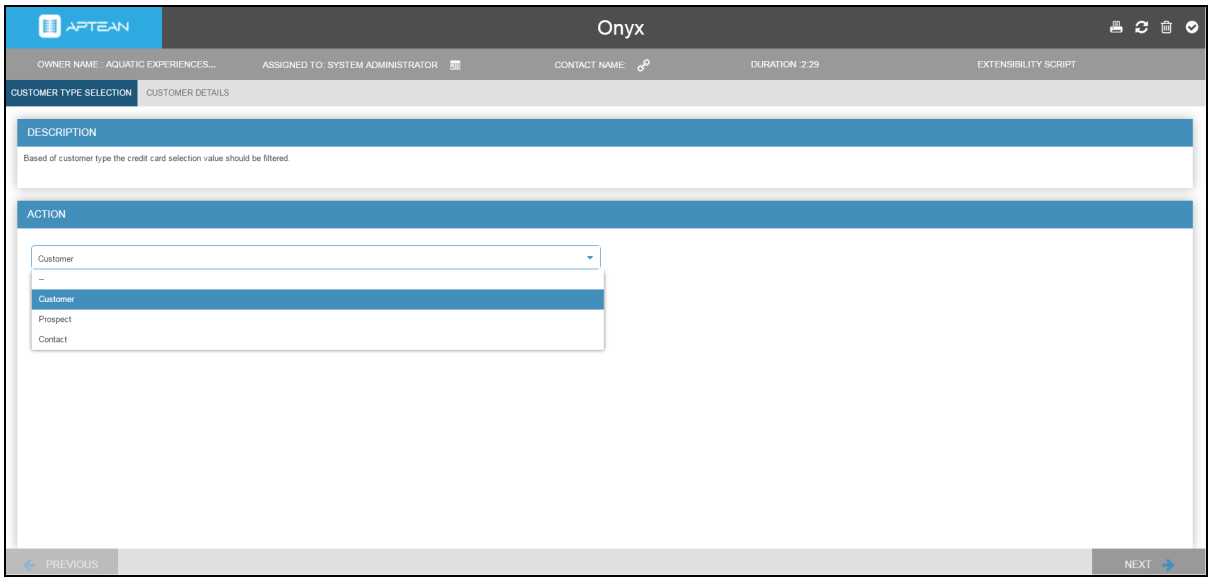


Figure 9-5: Customer type

Step 2: Based on the injected custom code, the Credit Card type appears as shown in the following image.

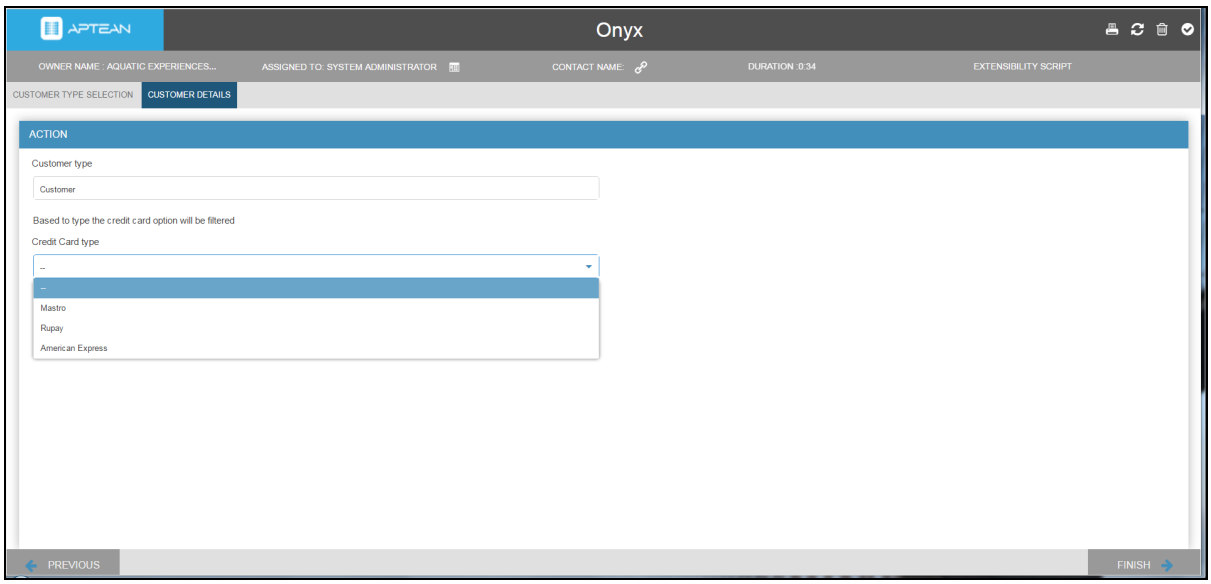


Figure 9-6: Credit Card type

Scripting configurations for extensibility

The base layout and component design for the script is as follows:

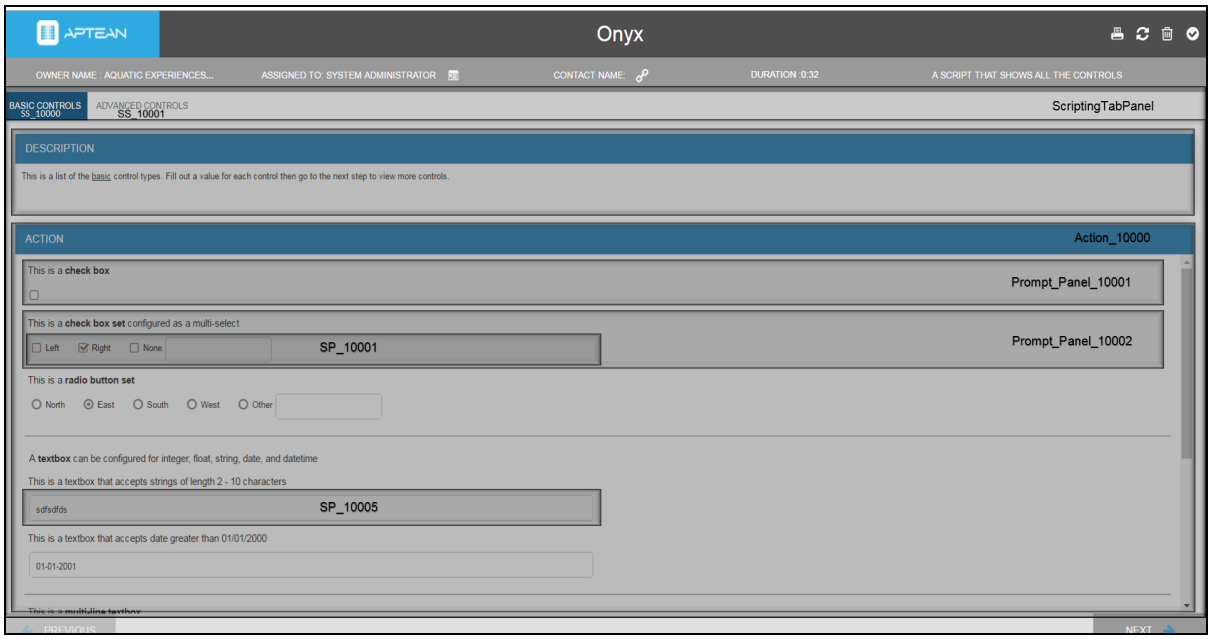


Figure 9-7: Base layout and component design

The component is divided in a modular manner with all major component set with an Id:

- `Scripting.internal.ScriptIdentifierConstants.SCRIPT_CENTER_CONTAINER = "ScriptingTabPanel"` is the tab layout. Where, each tab represents a script step.
- `Scripting.internal.ScriptIdentifierConstants.SCRIPT_STEP_PREFIX = "SS_"` is the prefix to the Id set to the panel that contains the Script Step view. SS_ is concatenated with the Script Step Id generated by the script designer. For example, consider Script Step Id generated by the script designer is **100000**, then the Id to the Script Step will be set as **SS_100000**.
- `Scripting.internal.ScriptIdentifierConstants.STEP_ACTION_PANEL_PREFIX = "Action_"` is the prefix to the Id set to the panel that contains all the Script Prompts (controls). These are on Action panel for each step that is created. Action_ is concatenated with the Script Step ID generated by the script designer. For example, consider Script Step Id generated by the script designer is **100000**, then the Id to the Action panel will be set as **Action_ 100000**.
- `Scripting.internal.ScriptIdentifierConstants.SCRIPT_STEP_PROMPT_PREFIX = "SP_"` is the prefix to the Id set to the Script Prompts (controls). There can be 'n' number of controls in each Action panel for each Step Prompts that is created in the Script Desginer. SP_ is concatenated with the Step Prompt Id generated by the script designer for each

Prompts. For example, consider Step Prompt Id generated by the script designer is **100000**, then the Id to the Step Prompt will be set as **SP_100000**.

Scripting Prompts Implementation

Following are the implementations of Scripting Prompts:

On Lazy Loading Prompts

The On Lazy Loading Prompts are loaded on demand once the 'AfterRender' of all Sencha components are completed. This helps in loading huge data as you do not have to load the entire data into the application at once. This is done for improving the performance. These controls cannot be customized during the 'AfterRender' of the Sencha component. Hence, you are provided with an AfterLazyLoad method for its customization.

Following are the On Lazy Load Controls:

- Country
- Drop down list box
- Incident Product
- Reference lookup
- Region
- User
- Product
- Tracking code

Note: The User, Product, and Tracking code tree control methods are not customizable for the Onyx 7.8 release.

On Application Load Prompts

All the controls except On Lazy Load Controls fall under On Application Load Prompts category. These controls can be customized at 'AfterRender' method.

Following are the few exaples of On Application Load Controls:

- Text box
- Muliti line text box
- Check box

Server Configuration

Application configuration files contain settings specific to an application. This file contains configuration settings that the common language runtime reads, and settings that the application can read.

WebAPI provides the following configuration for the users :

1. `Unityui.config` – This is configuration for IOC Containers. All the controllers and business layer components dependencies are configured within the config file.
   This config contains the assembly configuration of Scripting BusinessLayer. The `OnyxUIContainer` container contains the `Onyx.UI.Scripting.Interface` and `Onyx.UI.Core.Interface` map to its class's. The default mapping of the Scripting libraries are done here.

2. `unitysdk.config` – This is a Unity Container. Unity is a lightweight, extensible dependency injection container. Unitysdk contains the assembly configuration of Onyx SDK. The `OnyxSDKContainer` container contains the `OnyxSDK.Public.Interface.OEASServices` to its class's. The OEASService interface is mapped by default to the OnyxOEASServices. You can also map the OEASService interface with OnyxOGSServices.
   In addition, `OnyxSDK.Public.Interface.EnterpriseService` is the EnterpriseService library configuration that the user can use.
3. `languageculturemapping.config` – This configuration helps to map Language Culture. This configuration file is used to map the culture information of the browser to the language of onyx. For example, En-US, En-UK can be mapped to same language in Onyx, English. This is used for the initial loading of the scripts before the log in process.
4. `ogssdkservice.config` – When the mapping of OEASService interface is to the `OnyxOGSServices` in the `unitysdk.config` then the `ogssdkservice.config` gives the connection details of the OGS service that user wants to point to.
5. `log4net.config` - log4net is a tool to help the programmer output log statements to a variety of output targets. In case of problems with an application, it is helpful to enable logging so that the problem can be located. With log4net it is possible to enable logging at run-time without modifying the application binary. The log4net package is designed so that log statements can remain in shipped code without incurring a high performance cost. It follows that the speed of logging (or rather not logging) is crucial.

The Scripting server application has dependency on the Onyx SDK, which can interact with OEAS or with OGS, which interacts with OEAS eventually.

When OnyxSDK points to OnyxOEASService, the following configuration is used:

```
<container name="OnyxSDKContainer">
    <register type="OnyxSDK.Public.Interface.OEASService.IBaseOEAS"
mapTo="OnyxSDK.Private.OEASService.OnyxOEASService">
      <lifetime type="transient" />
    </register>
    <register
type="OnyxSDK.Public.Interface.OEASService.IBaseOEASOnyxAuthenticat
e"
.
.
.
    <register
type="OnyxSDK.Public.Interface.EnterpriseService.IBaseES"
mapTo="OnyxSDK.Public.EnterpriseService.OnyxEnterpriseService">
      <lifetime type="transient" />
    </register>
    <register
type="OnyxSDK.Public.Interface.EnterpriseService.IBaseESOnyxAuthenti
cate"
.
.
.
  </container>
```

When OnyxSDK points to OnyxOGSService, the following configuration is used:

```
<container name="OnyxSDKContainer">

<register type="OnyxSDK.Public.Interface.OEASService.IBaseOEAS"
mapTo="OnyxSDK.Private.OGSService.OnyxOGSService">
      <lifetime type="transient" />
    </register>
.
.
.


    <register
type="OnyxSDK.Public.Interface.EnterpriseService.IBaseES"
mapTo="OnyxSDK.Public.EnterpriseService.OnyxEnterpriseService">
      <lifetime type="transient" />
    </register>
.
.
.
  </container>
```

In Addition, we have to specify the `OnyxSDKGatewayServiceConfigurationsSection` to point to the OGS service to be used:

```
<OnyxSDKGatewayServiceConfigurationsSection>
  <OnyxSDKGatewayServiceConfigurations>
    <OnyxSDKGatewayServiceConfiguration UniqueId="" BaseAddress=""
DomainName="" UserName="" Password="" AuthenticationType=""
Application="" Site="" Language=""/>
  </OnyxSDKGatewayServiceConfigurations>
</OnyxSDKGatewayServiceConfigurationsSection>
```

Scripting Server Controller Extensibility

All the scripting related controllers in the scripting server controller is present in the assembly:

- Onyx.UI.Scripting.Controller.dll
- Onyx.UI.Survey.Controller.dll

The business layer interfaces and components are located in assemblies:

- Onyx.UI.Scripting.Interface.dll
- Onyx.UI.Scripting.BusinessLayer.dll
- Onyx.UI.Survey.Interface.dll
- Onyx.UI.Survey.BusinessLayer.dll

The business layer components implement most of the interfaces listed in the interface assembly. When the application requests for the instance of any of the interface assembly through Unity, it checks in the `unityui.config` file to return back the instance. If any change required in the business layer, a new assembly required to implement the interface and new class need to be registered in the `unityui.config` file. The configuration is loaded into application memory so dynamic changes to the config file are not reflected, so restart of the application is needed.