Oracle Endeca Experience Manager Extensions

User's Guide

Version 2.1.2 • November 2012 • Revision A

Contents

Copyright and disclaimer

Preface

The Oracle Endeca Commerce solution enables your company to deliver a personalized, consistent customer buying experience across all channels — online, in-store, mobile, or social. Whenever and wherever customers engage with your business, the Oracle Endeca Web commerce solution delivers, analyzes, and targets just the right content to just the right customer to encourage clicks and drive business results.

Oracle Endeca Commerce is the most effective way for your customers to dynamically explore your storefront and find relevant and desired items quickly. An industry-leading faceted search and Guided Navigation solution, Oracle Endeca Commerce enables businesses to help guide and influence customers in each step of their search experience. At the core of Oracle Endeca Commerce is the MDEX Engine™, a hybrid search-analytical database specifically designed for high-performance exploration and discovery. The Endeca Content Acquisition System provides a set of extensible mechanisms to bring both structured data and unstructured content into the MDEX Engine from a variety of source systems. Endeca Assembler dynamically assembles content from any resource and seamlessly combines it with results from the MDEX Engine.

Oracle Endeca Experience Manager is a single, flexible solution that enables you to create, deliver, and manage content-rich, cross-channel customer experiences. It also enables non-technical business users to deliver targeted, user-centric online experiences in a scalable way — creating always-relevant customer interactions that increase conversion rates and accelerate cross-channel sales. Non-technical users can control how, where, when, and what type of content is presented in response to any search, category selection, or facet refinement.

These components — along with additional modules for SEO, Social, and Mobile channel support — make up the core of Oracle Endeca Experience Manager, a customer experience management platform focused on delivering the most relevant, targeted, and optimized experience for every customer, at every step, across all customer touch points.

About this guide

This guide describes the major tasks involved in installing Experience Manager Extensions and using the included extensions to Experience Manager and the Content Assembler API.

This guide assumes that you have read the *Oracle Endeca Experience Manager Developer's Guide* and the *Content Assembler API Developer's Guide* and that you are familiar with implementing and maintaining Experience Manager-driven applications. If you intend to use Experience Manager Extensions to develop your own custom editors, this guide assumes you are familiar with the Experience Manager Editor SDK as documented in the *Oracle Endeca Experience Manager Developer's Guide*.

Remember: All documentation is available on the Oracle Technology Network (OTN)..

Who should use this guide

This guide is intended for developers who are building Endeca applications using Experience Manager and the Content Assembler API, and who wish to use Experience Manager Extensions as an additional

development resource, or take advantage of the features enabled by the sample extensions included in Experience Manager Extensions.

Conventions used in this guide

This guide uses the following typographical conventions:

Code examples, inline references to code elements, file names, and user input are set in `monospace` font. In the case of long lines of code, or when inline monospace text occurs at the end of a line, the following symbol is used to show that the content continues on to the next line: ¬

When copying and pasting such examples, ensure that any occurrences of the symbol and the corresponding line break are deleted and any remaining space is closed up.

Contacting Oracle Support

Oracle Support provides registered users with important information regarding Oracle Endeca software, implementation questions, product and solution help, as well as overall news and updates.

You can contact Oracle Support through Oracle's Support portal, My Oracle Support at *https://support.oracle.com*.

Chapter 1

Installing Experience Manager Extensions

Experience Manager Extensions provides a set of cartridges to extend the functionality of Experience Manager and the Content Assembler API. It includes a collection of resources for developing Experience Manager extensions and a Flash Builder project for building Experience Manager community editors.

About Experience Manager Extensions

Experience Manager Extensions features a set of editors and associated tag handlers that expand the functionality of Experience Manager and the Content Assembler API. It also includes templates and sample rendering code to demonstrate the new editor features and a Flash Builder source project that allows you to customize and rebuild the included samples.

Some of the Experience Manager Extensions cartridges are intended to replace the cartridges included with the Content Assembler reference application, while others provide entirely new functionality. The rendering code samples provided with the replacement cartridges are backward-compatible with the standard Experience Manager editors and reference cartridges.

Source code is provided for all Experience Manager Extensions components, enabling you to modify and rebuild the cartridge components, or to create your own.

Prerequisites

This section describes the software requirements for installing and using Experience Manager Extensions.

Endeca software requirements

Experience Manager Extensions requires that you have the following Endeca components installed:

- MDEX Engine
- Platform Services
- Deployment Template
- Endeca Workbench with Experience Manager
- Content Assembler API for Java or Content Assembler API for RAD.NET

For version compatibility information, refer to the *Oracle Endeca Commerce Compatibility Matrix*.

Other software requirements

If you want to update files in the Java Content Repository that is included with Experience Manager Extensions, you may find it useful to have a WebDAV client, such as EngInSite DataFreeway.

WebDAV is supported by the "My Network Places" feature on Windows and the cadaver utility on UNIX (also available for Cygwin). If you have difficulty with WebDAV using "My Network Places" on 64-bit Windows, you should download a separate WebDAV client.

Related Links

About the Java Content Repository on page 19

The Experience Manager Extensions installation includes a JSR-170 compliant Java Content Repository (JCR) that manages several resources that enable the sample editors. This JCR instance is hosted within the Experience Manager Extensions Web application and can be browsed via WebDAV.

About installing Experience Manager Extensions

Experience Manager Extensions is packaged as a ZIP archive. It includes a WAR file for easily deploying the samples inside of your Workbench installation.

It can also be deployed from the included standalone JAR file, or by building the source project in a Flex environment.

Package contents

Experience Manager Extensions is packaged as a ZIP archive. You can extract it anywhere on your system. This topic describes the files created in the Endeca Experience Manager Extensions installation.

The ZIP archive contains the following files:

File / Directory	Contents
`/doc`	This guide, as well as the *Oracle Endeca Commerce Third-Party Software Usage and Licenses*. Additional documentation is available from the Oracle Technology Network (OTN)..
`/reference/PbxRefApp`	Experience Manager Extensions-enabled Content Assembler reference application for Java.
`/reference/PbxRefApp.xml`	The sample context file for the Experience Manager Extensions reference application for Java.
`/reference/PbxRefAppDotNet`	Experience Manager Extensions-enabled Content Assembler reference application for the RAD Toolkit for ASP.NET.
`/reference/pbx_wine_app`	The Experience Manager Extensions Wine reference application data and Deployment Template module, which includes the zones used by the Dynamic Slot Editor.
`/reference/thumbnails`	The editor thumbnails Web application. This is deployed alongside Workbench in order to provide editor thumbnail images.

File / Directory	Contents
/reference/thumbnails.xml	The context definition file for the editor thumbnails Web application.
/reference/pbx.xml	The sample context definition file for the Web application hosting Experience Manager Extensions editors.
oexmx-<version>-content.war	The included Web archive can be installed into the Endeca Tools Service and configured to run alongside Workbench.
oexmx-<version>-standalone.jar	The standalone JAR allows you to install Experience Manager Extensions as a standalone Web server that can be run independently of Endeca Workbench.
oexmx-<version>-source.zip	The source archive contains the files and directories required to set up the Experience Manager Extensions project in Flash Builder.

Tag handlers for the Content Assembler API are installed as part of the reference applications:

- Tag handlers for Java are packaged in /reference/PbxRefApp/WEB-INF/lib/xtags-<version>.jar.
- Tag handlers for the RAD Toolkit for ASP.NET are packaged in Endeca.Data.Content.Pbx.dll located in reference\PbxRefAppDotNet\bin.

The source code for the tag handlers is also provided within the Experience Manager Extensions source archive.

Source archive contents

The pbx-<version>-source.zip archive contains the files and directories required to set up the Experience Manager Extensions project in Flash Builder.

The pbx-<version>-source.zip includes the resources below:

Directory	Contents
antlib	Custom Ant task files for uploading content to a Sling repository.
aslib	Common ActionScript classes used in Experience Manager Extensions editors.
content	PBX-related content that sits on top of the reference application.
editors	Experience Manager Extensions editors that are loaded into Experience Manager.
libs	Build dependencies.
META-INF	Information regarding this distribution of the PBX source.
xtags	Tag handlers for Java required to support Experience Manager Extensions editors.
xtagsdotnet	Tag handlers for the RAD Toolkit for ASP.NET required to support Experience Manager Extensions editors.

Directory	Contents
`build.xml`	Ant build file for building and deploying Experience Manager Extensions from source.

About the Experience Manager Extensions application

The Experience Manager Extensions application contains the sample editors, as well as the configuration and resources they rely on.

Installing Experience Manager Extensions as a Web application

You can install the WAR version of the Extensions in the Endeca Tools Service to run the sample editors alongside Experience Manager. This is not recommended for a production environment.

To install and deploy Experience Manager Extensions from the WAR:

1. Stop the Endeca Tools Service.
2. Extract the Experience Manager Extensions ZIP into the directory of your choice, such as `C:\Endeca\oexmx`.
3. Copy the `\reference\pbx.xml` file from the extracted archive into the `%ENDECA_TOOLS_CONF%\conf\Standalone\localhost` directory (on Windows) or the `$ENDECA_TOOLS_CONF/conf/Standalone/localhost` directory (on UNIX).
4. Open the `pbx.xml` file.
5. Replace `@@PBX_INSTALL_DIRECTORY@@` with the path to the directory you created in Step 2, for example:

   ```
   <!--  Context configuration file for PBX (Page Builder Extensions)-->
   <Context path="/oexmx" docBase="C:\Endeca\oexmx\oexmx-2.1.2/oexmx-2.1.2-content.war" debug="0" privileged="false" />
   ```

6. Save and close the file.
7. Navigate to `%ENDECA_TOOLS_CONF%\conf` (on Windows) or `$ENDECA_TOOLS_CONF/conf` (on UNIX).
8. Open the `webstudio.properties` file.
9. Find the line that specifies the location of the editor configuration file, for example:

   ```
   # The URL from which to load editor configuration for Page Builder
   #com.endeca.webstudio.pagebuilder.editors.config=http://my.serv¬
   er.com/some/location/myconfiguration.xml
   ```

10. Uncomment and change it to point to the URL of the `pbconfig` configuration file, for example:

    ```
    com.endeca.webstudio.pagebuilder.editors.config=/oexmx/etc/pbconfig.xml
    ```

11. Save and close the file.
12. Start the Endeca Tools Service.

You can verify your installation by navigating to `http://<workbench-host>:<port>/oexmx/etc/pbconfig.xml` in your browser.

Installing Experience Manager Extensions as a standalone Web server

You can deploy Experience Manager Extensions from the included JAR file.

Deploying from the JAR runs Experience Manager Extensions on a separate Web server. This is recommended primarily for developers who want to deploy the sample editors independently of Endeca Workbench services.

To install Experience Manager Extensions from the JAR:

1. Extract the Experience Manager Extensions ZIP into the directory of your choice, such as `C:\Endeca\oexmx\oexmx-<version>`.
2. Open a command shell and navigate to the directory you created in Step 1.
3. Select an unused port and execute the JAR file with the following command:
   ```
   java -jar oexmx-<version>-standalone.jar -p <port>
   ```
4. Stop the Endeca Tools Service.
5. Navigate to `%ENDECA_TOOLS_CONF%\conf` (on Windows) or `$ENDECA_TOOLS_CONF/conf` (on UNIX).
6. Open the `webstudio.properties` file.
7. Find the line that specifies the location of the editor configuration file, for example:
   ```
   # The URL from which to load editor configuration for Page Builder
   #com.endeca.webstudio.pagebuilder.editors.config=http://my.serv¬
   er.com/some/location/myconfiguration.xml
   ```
8. Uncomment and change it to point to the absolute URL of the `pbconfig` configuration file, for example:
   ```
   com.endeca.webstudio.pagebuilder.editors.config=http://<host>:<port>/etc/pb¬
   config.xml
   ```
9. Save and close the file.
10. Start the Endeca Tools Service.

You can verify your installation by navigating to `http://<host>:<port>/etc/pbconfig.xml` in your browser.

Deploying the MDEX Engine data service application

The cartridges included with Experience Manager Extensions rely on the MDEX Engine data service to query the MDEX Engine.

This section assumes that you have installed the Experience Manager Extensions Web application in the Endeca Tools Service.

To deploy the MDEX Engine data service application:

1. Stop the Endeca Tools Service.
2. Open the `MdexService.xml` file.
3. Replace `@@PBX_INSTALL_DIRECTORY@@` with the path to the Experience Manager Extensions installation directory, for example:
   ```
   <Context
       path="/MdexService"
   ```

```
    docBase="C:\Endeca\oexmx-<version>/MdexService"
    debug="0"
    privileged="false"
/>
```

4. Save and close the file.
5. Copy the `MdexService.xml` file into the `%ENDECA_TOOLS_CONF%\conf\Standalone\localhost` directory (on Windows) or the `$ENDECA_TOOLS_CONF/conf/Standalone/localhost` directory (on UNIX).

   By default, this is `C:\Endeca\workspace\conf\Standalone\localhost` on Windows, or `/usr/local/endeca/workspace/conf/Standalone/localhost` on UNIX.
6. If your Dgraph is not running on the default port of 15000, you must update the deployment descriptor file located in `MdexService/WEB-INF/web.xml`.
7. Navigate to the `<install dir>/MdexService/WEB-INF/lib` directory.
8. Rename `javax.servlet-api-3.0.1.jar` to `javax.servlet-api-3.0.1.jad`

   This prevents the application server's classloader from reading the JAR file.
9. Start the Endeca Tools Service.

Deploying the metadata service application

The metadata service provides information to the Experience Manager Extensions cartridges.

This section assumes that you have installed the Experience Manager Extensions Web application in the Endeca Tools Service.

To deploy the metadata service Web application:

1. Stop the Endeca Tools Service.
2. Open the `MetaDataService.xml` file.
3. Replace `@@PBX_INSTALL_DIRECTORY@@` with the path to the Experience Manager Extensions installation directory, for example:

```
<Context
    path="/MetaDataService"
    docBase="C:\Endeca\oexmx-<version>/MetaDataService"
    debug="0"
    privileged="false"
/>
```

4. Save and close the file.
5. Copy the `MetaDataService.xml` file into the `%ENDECA_TOOLS_CONF%\conf\Standalone\localhost` directory (on Windows) or the `$ENDECA_TOOLS_CONF/conf/Standalone/localhost` directory (on UNIX).
6. If your Dgraph is not running on the default port of 15000, you must update the deployment descriptor file located in `MetaDataService/WEB-INF/web.xml`.
7. Navigate to the `<install dir>/MetaDataService/WEB-INF/lib` directory.
8. Rename `javax.servlet-api-3.0.1.jar` to `javax.servlet-api-3.0.1.jad`

   This prevents the application server's classloader from reading the JAR file.
9. Start the Endeca Tools Service.

Deploying the enhanced sample wine application

To configure the sample wine application, run the Deployment Template `deploy` script using the default options and use the included control scripts to initialize your application.

This section assumes that you have installed the Experience Manager Extensions Web application in the Endeca Tools Service.

To configure and deploy the enhanced sample wine application:

1. Use the Deployment Template to provision your application:
   a) Navigate to the `C:\Endeca\Solutions\deploymentTemplate-<version>\bin` directory on Windows or `/usr/local/Endeca/Solutions/deploymentTemplate-<version>/bin` on UNIX and run `deploy.bat` or `deploy.sh` with the following options:

      ```
      --app <install dir>/reference/wine_app/deploy.xml
      ```

      This script creates the project directories and configuration files.
   b) Enter information as prompted.

      The instructions in this guide assume an installation that uses the default settings and is integrated with your Workbench installation.

      Note: If you do not use port 15000 as your Dgraph port, you must update the MDEX Engine service definition with the appropriate host and port information.

2. Navigate to the `/control` subdirectory of your enhanced wine application.

   This is under the application directory you specified in Step 1b.
3. Run the `initialize_services.bat` or `initialize_services.sh` script.
4. Run the `load_baseline_test_data.bat` or `load_baseline_test_data.sh` script.
5. Run the `baseline_update.bat` or `baseline_update.sh` script.
6. Run the `set_templates.bat` or `set_templates.sh` script.

   The `get_templates` and `set_templates` batch files and shell scripts are part of the enhanced sample wine application. They are not included in the standard wine reference application.

   Note: If you make modifications to the templates in the `oexmx-source-<version>.zip` source archive under the `content/templates/` directory, you must copy the modified templates to the `<appDir>/page_builder_templates` directory before running `set_templates`.

You can verify your installation by navigating to `http://<host>:<port>/endeca_jspref` in your browser.

For additional details on using the Deployment Template, please see the *Oracle Endeca Experience Manager Getting Started Guide*.

Related Links

About the MDEX Engine service definition on page 26

The MDEX Engine service definition file enables you to configure communication information for your MDEX Engine. This information is used by the microbrowser and any editors that contain it.

Deploying the Experience Manager Extensions reference application for Java

You can install the Experience Manager Extensions reference application into the Endeca Tools Service by using the sample context file included in the installation package.

To deploy the Experience Manager Extensions reference application:

1. Stop the Endeca Tools Service.
2. Navigate to the `reference` subdirectory of your Experience Manager Extensions installation directory.
   For example: `C:\Endeca\oexmx\<version>\reference`.
3. Open the `PbxRefApp.xml` file.
4. Replace `@FULL.PATH.TO.PBX_REFAPP@` with the path to the Experience Manager Extensions reference application directory, for example:

```
<Context
    path="/PbxRefApp"
    docBase="C:\Endeca\oexmx-<version>\reference\PbxRefApp"
    debug="0"
    privileged="false"
/>
```

5. Save and close the file.
6. Copy the `PbxRefApp.xml` file into the `%ENDECA_TOOLS_CONF%\conf\Standalone\localhost` directory (on Windows) or the `$ENDECA_TOOLS_CONF/conf/Standalone/localhost` directory (on UNIX).
7. If your Dgraph is not running on the default port of 15000, you must update the deployment descriptor file located in `reference/PbxRefApp/WEB-INF/web.xml`.
8. Start the Endeca Tools Service.

You can verify the deployment by navigating to `http://<host>:<port>/PbxRefApp` in your browser, where *<host>* is the host name or IP address of the server running the reference application, and *<port>* is the Endeca Tools Service port (default 8006).

Deploying the Experience Manager Extensions reference application for the RAD Toolkit for ASP.NET

You can install the Experience Manager Extensions reference application into an instance of IIS 6 or 7.

This procedure is based on deploying the application in IIS 6. Some steps may vary for IIS 7.

To deploy the Experience Manager Extensions reference application:

1. If your Dgraph is not running on the default port of 15000, you must update the `Web.config` located in `reference\PbxRefAppDotNet`.

```
<endeca>
 <!-- additional elements omitted from this example -->
 <servers>
  <clear />
  <add name="Local" hostName="localhost" port="15000" certificatePath=""
```

```
/>
     </servers>
</endeca>
```

2. Open IIS Manager.
3. In the **Connections** pane, navigate to **Websites**, then **Default Web Site**.
4. Right-click on **Default Web Site** and select **New** > **Virtual Directory**.
5. Specify the values for the application.

Property	Description
Alias	A unique path name for your application, for example, `PbxRefApp`.
Path	The path to the application folder, for example, `C:\Endeca\oexmx-<version>\reference\PbxRefAppDotNet`

6. Right-click on your new application and select **Properties**.
7. On the **Virtual Directory** tab, click **Create**.
8. In the **Execute permissions** field, select **Script Only**.
9. Click **Configuration**, then **Insert**.
10. In the **Executable** field, specify the path to the ISAPI filter, for example, `C:\WINDOWS\Microsoft.NET\Framework64\v2.0.50727\aspnet_isapi.dll` and clear the checkbox.

You can verify the deployment by navigating to `http://<host>/<alias>` in your browser, where *<host>* is the host name or IP address of the server running the reference application, and *<alias>* is the value that you specified in Step 3.

About configuring tag handlers for the RAD Toolkit for ASP.NET

The reference application for the RAD Toolkit for ASP.NET includes the configuration for the sample tag handlers for .NET.

If you wish to use the tag handlers provided with Experience Manager Extensions in your own application, you can copy the sample configuration to your own application.

After copying `Endeca.Data.Content.Pbx.dll` to the `bin` directory of your application, include the following in your `Web.config` or `App.config`:

```
   <endeca.content>
      <tagHandlers>
         <add handlerType="Endeca.Data.Content.Pbx.BoostBuryResultsProces¬
sorTagHandler, Endeca.Data.Content.Pbx" />
         <add handlerType="Endeca.Data.Content.Pbx.DimensionSearchConfig¬
TagHandler, Endeca.Data.Content.Pbx" />
         <add handlerType="Endeca.Data.Content.Pbx.DynamicSlotTagHandler,
 Endeca.Data.Content.Pbx" />
        <add handlerType="Endeca.Data.Content.Pbx.NavigationRefinementCon¬
figTagHandler, Endeca.Data.Content.Pbx" />
        <add handlerType="Endeca.Data.Content.Pbx.NavigationStateTagHandler,
 Endeca.Data.Content.Pbx" />
        <add handlerType="Endeca.Data.Content.Pbx.RecordSelectionTagHandler,
 Endeca.Data.Content.Pbx" />
        <add handlerType="Endeca.Data.Content.Pbx.RecordSelectorTagHandler,
 Endeca.Data.Content.Pbx" />
         <add handlerType="Endeca.Data.Content.Pbx.RefinementDimensionCon¬
```

```
figsTagHandler, Endeca.Data.Content.Pbx" />
      </tagHandlers>
      <schemas>
         <add namespace="http://endeca.com/schema/content/xtags/2010"
uri="Endeca.Data.Content.Pbx.Resources.content-tags-2010.xsd" assembly="En¬
deca.Data.Content.Pbx"/>
      </schemas>
   </endeca.content>
```

Verifying your Experience Manager Extensions installation

You can verify your Experience Manager Extensions installation by confirming that the sample editors are available in Experience Manager.

To access the sample editors:

1. Log in to Endeca Workbench.
2. Select **Experience Manager**.
   The **Group View** displays.
3. Select **NavigationPageGroup**.
   The **List View** displays.
4. Open the **Default Landing Page**.
5. From the **Content Tree** on the left, select **LeftColumn**.
6. Click **Add**.
7. Select the **ThreeRecordBox** cartridge.
8. Click **Edit Query**.
   If your installation is configured correctly, the microbrowser displays.

If the editor does not display, check the configuration of the editor configuration file in `webstudio.properties`.

If the editor displays but does not load data from the MDEX Engine, check the MDEX Engine service definition and verify that you have enabled access to MDEX Web services.

Related Links

About the MDEX Engine service definition on page 26
The MDEX Engine service definition file enables you to configure communication information for your MDEX Engine. This information is used by the microbrowser and any editors that contain it.

Setting up access to MDEX Engine Web services from a Flex client on page 31
By default, MDEX Engine Web services are accessible from Experience Manager only if the MDEX Engine and Oracle Endeca Workbench are hosted on the same domain.

Installing Experience Manager Extensions as a Web application on page 12
You can install the WAR version of the Extensions in the Endeca Tools Service to run the sample editors alongside Experience Manager. This is not recommended for a production environment.

Installing Experience Manager Extensions as a standalone Web server on page 13
You can deploy Experience Manager Extensions from the included JAR file.

About the Java Content Repository

The Experience Manager Extensions installation includes a JSR-170 compliant Java Content Repository (JCR) that manages several resources that enable the sample editors. This JCR instance is hosted within the Experience Manager Extensions Web application and can be browsed via WebDAV.

If you installed Experience Manager Extensions as a Web application (using Endeca Tools Service on the default port), the root URL of the repository is `http://<workbench-host>:8006/pbx`. If you installed Experience Manager Extensions as a standalone Web server, the repository is located at the root of the Web server with the port that you specified when running the JAR file.

Within this content root directory, the Experience Manager Extensions artifacts are contained in the following subdirectories:

Directory	Description
/apps	Contains the `xml.jsp` file that generates the Experience Manager editor configuration file (`pbconfig.xml`). Typically, you manage this file within the Flash Builder project provided.
/etc	JCR system directory. Do not modify.
/images	By default, the Image and Image Map editors look for images in this location (this can be configured in the image server service definition). Several sample images are included with the Experience Manager Extensions installation. Oracle recommends that you add any images to within this directory of the repository.
/modules	Contains the compiled SWF for Experience Manager Extensions sample editors.
/services	Contains the service definitions for the MDEX Engine and image server that are used by the sample editors. Also contains the zone configuration file that is used for dynamic slots.
/templates	Contains a copy of the Experience Manager Extensions sample templates. **Note:** Experience Manager does not access the templates from this location. In order to enable templates within Experience Manager, upload them to Workbench using the emgr_update utility.
/var	JCR system directory. Do not modify.

The root directory of the repository also contains several files which are not used by the Experience Manager Extensions sample editors. You can disregard these.

This repository structure, aside from the system directories, is mirrored within the `content` directory contained in the Experience Manager Extensions source archive. If you are working with the provided Flash Builder project, the easiest way to manage these files is to update them locally in this location. The Ant deploy task provided with the project updates the repository with any files you may have created or updated in the these directories within `content`, such as the `xml.jsp` and any new or modified editors. The Ant task does not create any new directories, but if you create the corresponding directory in the repository directly, the Ant task pushes those files up to the repository as well.

If your changes do not require code changes (such as adding images or updating service definitions) and you are not managing these files through the Flex project, you can update the repository directly through WebDAV. You can also use WebDAV to create new directories to store additional resources as needed.

Chapter 2

Working With the Basic Cartridges

This section provides details about the basic cartridges included in Experience Manager Extensions, including how to use and configure them.

About the basic cartridges

Experience Manager Extensions includes several sample cartridges designed to work with the provided sample wine application.

The **ThreeRecordBox** cartridge and the **ImageBox** cartridge are designed to replace the standard cartridges in the Content Assembler reference application to provide an improved method of browsing for and selecting records and images for landing pages. The **ImageMap** cartridge combines the improved browsing of images with the ability to specify links associated with image hotspots.

These cartridges include sample editors built around the new microbrowser and image browser components. These components make use of service definition files for configuring communication with an MDEX Engine or image repository.

In addition, the Extensions includes several basic sample editors, including a checkbox, color selector, drop-down menu, rich text editor, and string value editor. These are primarily intended as generic components for use in creating more advanced editor interfaces.

Related Links

Experience Manager Extensions basic cartridge reference on page 22
Experience Manager Extensions includes several basic cartridges for use with Experience Manager and a Content Assembler–enabled application.

About the microbrowser on page 26
The microbrowser is used in several of the editors in Experience Manager Extensions, including the Advanced Record List editor, Link editor, and Image Map editor. It communicates with the MDEX Engine specified in the MDEX Engine service definition file.

About the image browser on page 35
The image browser enables a content administrator to select images by browsing a content management system. It is a main component of the Image and Image Map editors.

Experience Manager Extensions basic cartridge reference

Experience Manager Extensions includes several basic cartridges for use with Experience Manager and a Content Assembler–enabled application.

Each cartridge is made up of several components:

Sample template	Located in `<app dir>/config/page_builder_templates` within your deployed sample wine application.
Editor	A compiled bundle of Experience Manager Extensions editors is installed into the JCR repository at `<content root>/modules/editors.swf`. The source code for the editors is also provided within the Experience Manager Extensions source archive.
Tag handler	Tag handlers for the Content Assembler API are installed as part of the reference applications for Java and for the RAD Toolkit for ASP.NET.
Renderer	In the reference application for Java, a sample renderer for each cartridge is provided in the `/reference/ContentAssemblerRefApp/cartridges` directory. In the reference application for the RAD Toolkit for ASP.NET, sample renderers are located in `reference\PbxRefAppDotNet\Re¬sources`.

Related Links

ThreeRecordBox cartridge on page 22
The Experience Manager Extensions version of the ThreeRecordBox cartridge replaces the standard ThreeRecordBox to enable content administrators to specify featured records or dynamic records using the microbrowser interface.

ImageBox cartridge on page 24
The Experience Manager Extensions version of the ImageBox cartridge replaces the standard ImageBox to enable content administrators to select an image using the image browser interface, and optionally specify a target link for the image.

ImageMap cartridge on page 25
The ImageMap cartridge is new in Experience Manager Extensions. It enables content administrators to select an image using the image browser interface and add hotspots to the image to create an image map.

ThreeRecordBox cartridge

The Experience Manager Extensions version of the ThreeRecordBox cartridge replaces the standard ThreeRecordBox to enable content administrators to specify featured records or dynamic records using the microbrowser interface.

The cartridge includes the following components:

Sample template	`SidebarItem-ThreeRecordBox.xml`
Editors	The sample cartridge uses a `RecordListEditor` to enable the microbrowser for selecting spotlighted records and a `StringEditor` to specify the display text for the "See all" link for dynamic records.

Tag handler	In Java, the `com.endeca.content.xtags.RecordSelectionTag` transforms the XML output of the `RecordListEditor` into a object of type `com.endeca.content.ene.RecordListProperty`. In the RAD Toolkit for ASP.NET, the `RecordSelectionTagHandler` transforms the XML output of the `IRecordListProperty` into an object of type `IRecordListProperty`
Renderer	`ThreeRecordBox.jsp` (for Java) and `ThreeRecordBoxSpotlight.ascx` (for the RAD Toolkit for ASP.NET).

Following is an example of the property and editor definitions in the template for the ThreeRecordBox cartridge:

```
<ContentItem>
  <Name>New Three Record Spotlight Box</Name>
  <Property name="title">
    <String>Featured Items</String>
  </Property>
  <Property name="record_list">
    <xtags:record_list/>
  </Property>
  <Property name="link_text">
    <String/>
  </Property>
</ContentItem>

<EditorPanel>
  <BasicContentItemEditor>
    <GroupLabel label="Configuration"/>
    <editors:StringEditor propertyName="title" label="Banner title"
      enabled="true"/>
    <editors:RecordListEditor propertyName="record_list"
      label="Featured records" maxRecords="3">
        <PreviewProperty name="P_Name" label="Name"/>
    </editors:RecordListEditor>
    <editors:StringEditor propertyName="link_text"
      label="See-all link text" enabled="true"/>
  </BasicContentItemEditor>
</EditorPanel>
```

The `<RecordListEditor>` uses the microbrowser to replace the standard record list support built into Experience Manager and the Content Assembler. In addition to the standard `propertyName` and `label` attributes, the `<RecordListEditor>` takes the following attributes:

Attribute	Description
`dataServiceId`	Specifies the location of the MDEX Engine service definition as the path to the service definition file, relative to the context root, omitting the `.xml` extension.

ImageBox cartridge

The Experience Manager Extensions version of the ImageBox cartridge replaces the standard ImageBox to enable content administrators to select an image using the image browser interface, and optionally specify a target link for the image.

Note: The ImageBox cartridge is only implemented in the reference application for Java and not for the RAD Toolkit for ASP.NET.

The cartridge includes the following components:

Sample template	`SidebarItem-ImageBox.xml`
Editor	The sample cartridge uses an `ImageEditor` to enable the image browser for selecting images from a content management system and a `LinkEditor` to specify the target link as a static URL or an Endeca navigation state.
Tag handler	In Java, the `com.endeca.content.xtags.ImageTag` handler transforms the XML output of `ImageEditor` into an object of type `com.endeca.content.xtags.model.Image`.
Renderer	`Image.jsp`

Following is an example of the property and editor definitions in the template for the ImageBox cartridge:

```
<ContentItem>
  <Name>Image Box</Name>
  <Property name="image">
    <Image xmlns="http://endeca.com/schema/content/xtags/2010"/>
  </Property>
  <Property name="caption">
    <String/>
  </Property>
  <Property name="link_reference">
    <Link xmlns="http://endeca.com/schema/content/xtags/2010">
      <target>_self</target>
    </Link>
  </Property>
</ContentItem>

<EditorPanel>
  <BasicContentItemEditor>
    <editors:ImageEditor propertyName="image" label="Image"/>
    <editors:StringEditor propertyName="caption" label="Caption"/>
    <editors:LinkEditor propertyName="link_reference"
      label="Page reference"/>
  </BasicContentItemEditor>
</EditorPanel>
```

The `<ImageEditor>` enables the image browser interface. In addition to the standard `propertyName` and `label` attributes, the `<ImageEditor>` takes the following attributes:

Attribute	Description
`imageServiceId`	Specifies the location of the image server service definition as the path to the service definition file, relative to the context root, omitting the `.xml` extension.

The <LinkEditor> enables a content administrator to specify a link to a static URL or a dynamic page based on a set of Endeca search results. In addition to the standard propertyName and label attributes, the <LinkEditor> takes the following attributes:

Attribute	Description
dataServiceId	Specifies the location of the MDEX Engine service definition as the path to the service definition file, relative to the context root, omitting the .xml extension.

The <LinkEditor> also enables a drop-down menu for the content administrator to specify the target attribute for the generated <link> element. You can specify the default value via the <target> element in the property definition. Valid values are:

Value	Description
_blank	Load in a new window
_self	Load in the same frame as it was clicked
_parent	Load in the parent frameset
_top	Load in the full body of the window

ImageMap cartridge

The ImageMap cartridge is new in Experience Manager Extensions. It enables content administrators to select an image using the image browser interface and add hotspots to the image to create an image map.

Note: The ImageMap cartridge is only implemented in the reference application for Java and not for the RAD Toolkit for ASP.NET.

The cartridge includes the following components:

Sample template	MainColumnContent-ImageMap.xml
Editor	The sample cartridge uses an ImageEditor with the image map editor functionality enabled.
Tag handler	The com.endeca.content.xtags.ImageTag handler transforms the XML output of ImageEditor into a Java object of type com.endeca.content.xtags.model.Image.
Renderer	Image.jsp

Following is an example of the property and editor definitions in the template for the ImageMap cartridge:

```
<ContentItem>
  <Name>Image Map</Name>
  <Property name="image">
    <Image xmlns="http://endeca.com/schema/content/xtags/2010"/>
  </Property>
  <Property name="map">
    <ImageMap xmlns="http://endeca.com/schema/content/xtags/2010"/>
  </Property>
```

```
  </ContentItem>

  <EditorPanel>
    <BasicContentItemEditor>
      <editors:ImageEditor propertyName="image" label="Image"
        mapPropertyName="map" enabled="true"/>
    </BasicContentItemEditor>
  </EditorPanel>
```

The `<ImageEditor>` enables the image browser interface, and can optionally enable the image map editor to enable a content administrator to create hotspots within an image that link to static pages or to a set of Endeca search results for a specified navigation state. In addition to the standard `propertyName` and `label` attributes, the `<ImageEditor>` takes the following attributes:

Attribute	Description
`imageServiceId`	Specifies the location of the image server service definition as the path to the service definition file, relative to the context root, omitting the `.xml` extension.
`mapPropertyName`	Specifies the name of a property (defined in the template) of type `ImageMap` that should contain a list of hotspot descriptions. If this attribute is present, the image map editor is enabled within the image browser. If no `mapPropertyName` is specified, the image map editor is not available for this cartridge.

About the microbrowser

The microbrowser is used in several of the editors in Experience Manager Extensions, including the Advanced Record List editor, Link editor, and Image Map editor. It communicates with the MDEX Engine specified in the MDEX Engine service definition file.

The microbrowser enables a content administrator to select records for spotlighting and is a replacement for the standard record selector in Experience Manager. This offers several advantages over the standard record list cartridge:

- It enables a content administrator to use traditional search and guided navigation to locate individual records, instead of record id lookup.
- It enables a content administrator to identify a dynamic record selection by specifying search terms, instead of or in addition to specifying dimension refinements.
- It features centralized configuration of an MDEX Engine data source, including the host and port of the MDEX Engine and display formatting for property and dimension values.

Note: The microbrowser depends on MDEX Engine 6.1.3 or 6.1.4 and the XQuery / Web service interface.

About the MDEX Engine service definition

The MDEX Engine service definition file enables you to configure communication information for your MDEX Engine. This information is used by the microbrowser and any editors that contain it.

The MDEX Engine service definition specifies:

- the server's location (host and port)
- the list of properties or dimensions to display in the results grid within the microbrowser.
- Optionally, you can specify an aggregation key to enable the microbrowser to operate in aggregated record mode.

A default MDEX Engine service definition is provided out of the box at `<content-root>/services/mdex.xml`. This default definition uses the Workbench MDEX Engine, and defines display properties suitable to the sample wine dataset. If you edit this configuration, all editors using that configuration file will be updated.

The MDEX Engine service definition file includes a root element, `<ServiceDefinition>` with the attribute `type="mdex"` and the following subelements:

Element	Usage
`<title>`	The service title. The microbrowser displays this to the user to identify the working data set.
`<host>`	Optional. The hostname or IP address of your MDEX Engine server. By default, the editor attempts to connect to the Workbench MDEX Engine.
`<port>`	Optional. The port that the MDEX Engine server listens on. By default, the editor attempts to connect to the Workbench MDEX Engine.
`<aggrega¬ tionKey>`	Optional. The property or dimension to use as the aggregation key when displaying and sorting records. All records with the same value in the selected dimension or property are treated as a single record.
`<wildcard¬ SearchEn¬ abled>`	Optional. Wildcard search is enabled by default. If your configuration does not index dimensions by wildcard index, you must explicitly set this property to `false`.
`<field>`	Each `<field>` element specifies a property or dimension to display in the microbrowser. Each field corresponds to a column in the results grid within the microbrowser interface. The `name` attribute of the `<field>` element is the name of the property or dimension in the MDEX Engine, such as "P_Price" or "Vintage". The first `<field>` listed displays in the **Record Selection** pane to indicate selected records in a cartridge that uses the Advanced Record List editor.
`<format>`	The `<format>` element is an optional subelement of `<field>`. It controls text formatting in the microbrowser on the values for the corresponding field. Valid values include the following: • `plain` — no formatting. Used as the default if no `<format>` element is present. • `currency` — adds a dollar ($) symbol before the value. • `integer` — removes the decimal point and any trailing digits, if present. This setting does not round the integer value. • `html` — attempts to handle markup tags within the content returned from the MDEX Engine.

Following is an example of an MDEX Engine service definition:

```
<ServiceDefinition type="mdex">
<title>Product Catalog | Spring 2010 | STAGING</title>
<!--  Workbench's main MDEX setting is used by default.  To override,
      uncomment the elements below and edit their values.
  <host>localhost</host>
  <port>15000</port>
-->
<!--  To specify an aggregation key, uncomment the element below and edit.

    <aggregationKey>P_Winery</aggregationKey>

-->
<!--  If Dimensions are not indexed by Wildcard Index, this property needs
 to be
      set to false.
  <wildcardSearchEnabled>false</wildcardSearchEnabled>
-->

<fields>
  <field name="P_Name" />
  <field name="P_Price">
    <format>currency</format>
  </field>
  <field name="P_Score">
    <format>integer</format>
  </field>
  <field name="P_SalesRank">
    <format>integer</format>
  </field>
  <field name="P_Winery">
    <format>html</format>
  </field>
  <field name="P_WineType" />
</fields>
</ServiceDefinition>
```

Related Links

Updating the service definitions and zone configuration file on page 37
the Experience Manager Extensions sample editors use service definition files that define how to communicate with external resources, such as an MDEX Engine or an image server.

Configuring multiple MDEX Engine data services

If your application data is stored across multiple MDEX Engine instances, you can specify a new MDEX Engine on a per-cartridge basis for any editors that use the microbrowser. First, create a data service that queries the desired MDEX Engine. Then, update the relevant cartridge template(s) to use the new data service.

Once your MDEX Engine service is configured and running correctly, you can create modified versions of it that each query an additional Dgraph. Ensure your data is loaded into each additional MDEX Engine, and the associated Dgraph process is running.

To configure additional MDEX Engine data services:

1. Navigate to your Experience Manager Extensions directory.

2. Copy the `oexmx-<version>/MdexService` directory to a new directory for the service you are creating.

   For example, create a copy of the directory as `oexmx-<version>/MdexService2`.

3. Modify the `web.xml` file for the new service:
   a) Open the `web.xml` file.

      For the above example, this would be `oexmx-2.1.2/MdexService2/WEB-INF/web.xml`.

   b) Locate the elements that specify the MDEX Engine host and port:

```
<init-param>
    <param-name>mdex-host</param-name>
    <param-value>localhost</param-value>
</init-param>
<init-param>
    <param-name>mdex-port</param-name>
    <param-value>15000</param-value>
</init-param>
```

   c) Modify the values to use the host and port of your new MDEX Engine:

```
<init-param>
    <param-name>mdex-host</param-name>
    <param-value>my.second.dgraph.host</param-value>
</init-param>
<init-param>
    <param-name>mdex-port</param-name>
    <param-value>15001</param-value>
</init-param>
```

   d) Save and close the file.

4. Create a modified MDEX Engine service definition file:
   a) Using a WebDAV client, retrieve the `<content root>/services/mdex.xml` file from the repository.
   b) Rename the file (for example, to `mdex2.xml`).
   c) Open the file and locate the `<host>` and `<port>` elements:

```
<ServiceDefinition type="mdex">
<title>Product Catalog | Spring 2010 | STAGING</title>
<host>WorkbenchHost.oracle.com</host>
<port>8006</port>
<serviceName>MdexService</serviceName>
<dimensionCountShowLimit>100</dimensionCountShowLimit>
[...]
```

   d) Modify the `<host>` and `<port>` values to those for your new MDEX Engine:

```
<ServiceDefinition type="mdex">
<title>Product Catalog | Spring 2010 | STAGING</title>
<host>my.second.dgraph.host</host>
<port>15001</port>
<serviceName>MdexService</serviceName>
<dimensionCountShowLimit>100</dimensionCountShowLimit>
[...]
```

   e) Modify the `<serviceName>` element to use the name of your new service.

      For the current example, this is `MdexService2`:

```
<ServiceDefinition type="mdex">
<title>Product Catalog | Spring 2010 | STAGING</title>
```

```
<host>my.second.dgraph.host</host>
<port>15001</port>
<serviceName>MdexService2</serviceName>
<dimensionCountShowLimit>100</dimensionCountShowLimit>
[...]
```

f) Save and close the file.

g) Upload the file to `<content root>/services` in the repository.

5. Create a context file for the new Web application within your Web application container:

   a) Navigate to the `%ENDECA_TOOLS_CONF%\conf\Standalone\localhost` directory (on Windows) or the `$ENDECA_TOOLS_CONF/conf/Standalone/localhost` directory (on UNIX).

   By default, this is `C:\Endeca\workspace\conf\Standalone\localhost` on Windows, or `/usr/local/endeca/workspace/conf/Standalone/localhost` on UNIX.

   b) Create a copy of the `MdexService.xml` file and rename it to use the name of your new data service.

   For the current example, this is `MdexService2.xml`.

   c) Open the file.

   d) Replace `docBase` with the path to the service directory you created in Step 2, for example:

```
<Context
    path="/MdexService"
    docBase="C:\Endeca\oexmx-<version>/MdexService2"
    debug="0"
    privileged="false"
/>
```

   e) Save and close the file.

6. Repeat the above steps for any additional MDEX Engine instances you wish to enable in the Experience Manager Extensions editors.

7. Update your cartridge templates to point to the appropriate service definition file.

   To point to the new MDEX Engine service definition file, modify the `dataServiceId` attribute in the editor configuration in the template, as shown below:

```
<editors:LinkEditor propertyName="link"
    dataServiceId="services/mdex2"/>
```

   The `dataServiceId` attribute is the path to the service definition file, relative to the context root, omitting the `.xml` extension.

8. Upload the new templates to your Workbench instance:

   a) Open a command prompt or UNIX shell.

   b) Run emgr_update with the `--action` of `set_templates` and the following parameters:

   - `--host` — The machine name and port for the Endeca Workbench environment, in the format *host*:*port*.
   - `--app_name` — The name of the application to which you want to add the templates.

   The following is a Windows example:

```
emgr_update.bat --action set_templates --host localhost:8006
--app_name My_application
```

   The following is a UNIX example:

```
emgr_update --action set_templates --host localhost:8006
--app_name My_application
```

You may wish to clear the cache of any web browser that previously accessed your Experience Manager instance, to ensure that you are not using cached data.

Related Links

Deploying the MDEX Engine data service application on page 13

The cartridges included with Experience Manager Extensions rely on the MDEX Engine data service to query the MDEX Engine.

Setting up access to MDEX Engine Web services from a Flex client

By default, MDEX Engine Web services are accessible from Experience Manager only if the MDEX Engine and Oracle Endeca Workbench are hosted on the same domain.

For example, if Endeca Workbench is hosted on `apps.example.com`, the MDEX Engine must also be accessible at `apps.example.com`, and Experience Manager extensions must be configured to access the MDEX Engine at `"apps.example.com"`. Using the host's IP address or an alias hostname, such as `"localhost"` causes a "Security Error" alert box to appear in Experience Manager when the editor attempts to access the MDEX Engine.

If the MDEX Engine is hosted on a different domain from Endeca Workbench, you must set up a cross-domain policy file on the MDEX Engine server. These steps apply to any Flex client application that communicates with an MDEX Engine via Web services.

To configure cross-domain access to MDEX Web services from a Flex client:

1. Navigate to the `/conf/dtd/xform` directory of your MDEX Engine installation, for example:
   `C:\Endeca\MDEX\6.1.4\conf\dtd\xform`
2. Create an Adobe Flash cross-domain policy file, `crossdomain.xml`.
3. Configure your `crossdomain.xml` file to grant access to all domains hosting instances of Endeca Workbench.
   An example is provided below:

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM "http://www.macrome¬
dia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
  <allow-access-from domain="*.example.com" />
  <allow-http-request-headers-from domain="*" headers="SOAPAction" />
</cross-domain-policy>
```

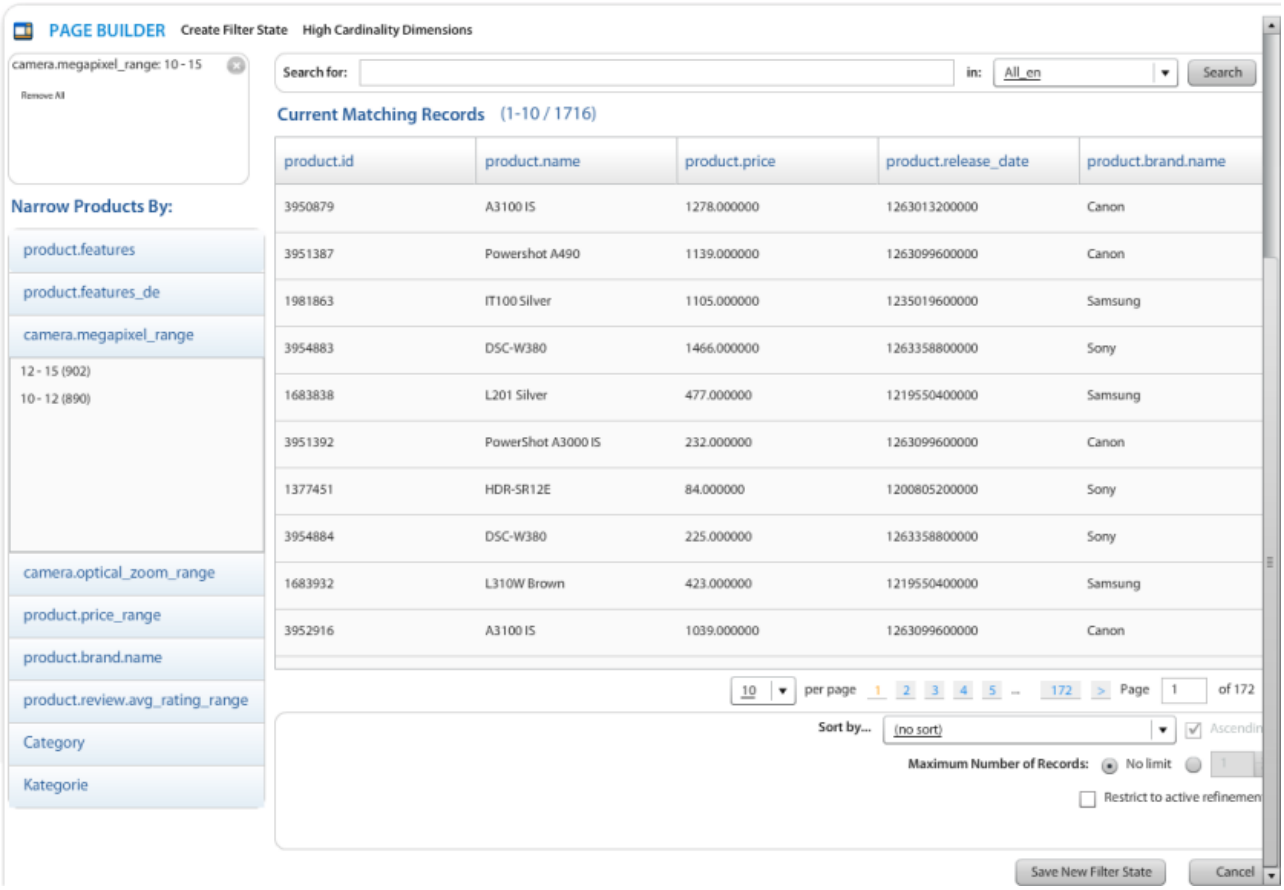
   - The `<allow-access-from>` element grants access to the local MDEX Engine Web service from a set of domains. The `domain` attribute may be specific, or may include a wildcard, as shown above. You can include any number of `<allow-access-from>` elements, each for a different domain.
   - The `<allow-http-request-headers-from>` element as specified above is required. It enables Flash clients to communicate with the MDEX using the SOAP protocol.

For a complete specification of the cross-domain policy file format, please see the Adobe documentation at *http://www.adobe.com/devnet/articles/crossdomain_policy_file_spec.html*.

About using the microbrowser in Experience Manager

The microbrowser enables a content administrator to specify a dynamic page link based on a set of navigation refinements.

The microbrowser UI is shown below in Create Filter State mode:



The UI consists of the following components:

- **Window Title** – This is specified in the MDEX Engine service definition, and identifies the working data set to the user.
- **Search for** field – The **Search for** field lets you select your dynamic record results page based on a set of search terms. You can combine it with the **in:** dropdown to limit your search to a specific MDEX Engine property or dimension. Selected search terms display in the **Filter State** pane.

  Note: The **Search for** field is typeahead-enabled. If a user selects one of the suggested dimensions, the filter state resets and the microbrowser adds the suggested dimension as the first breadcrumb.

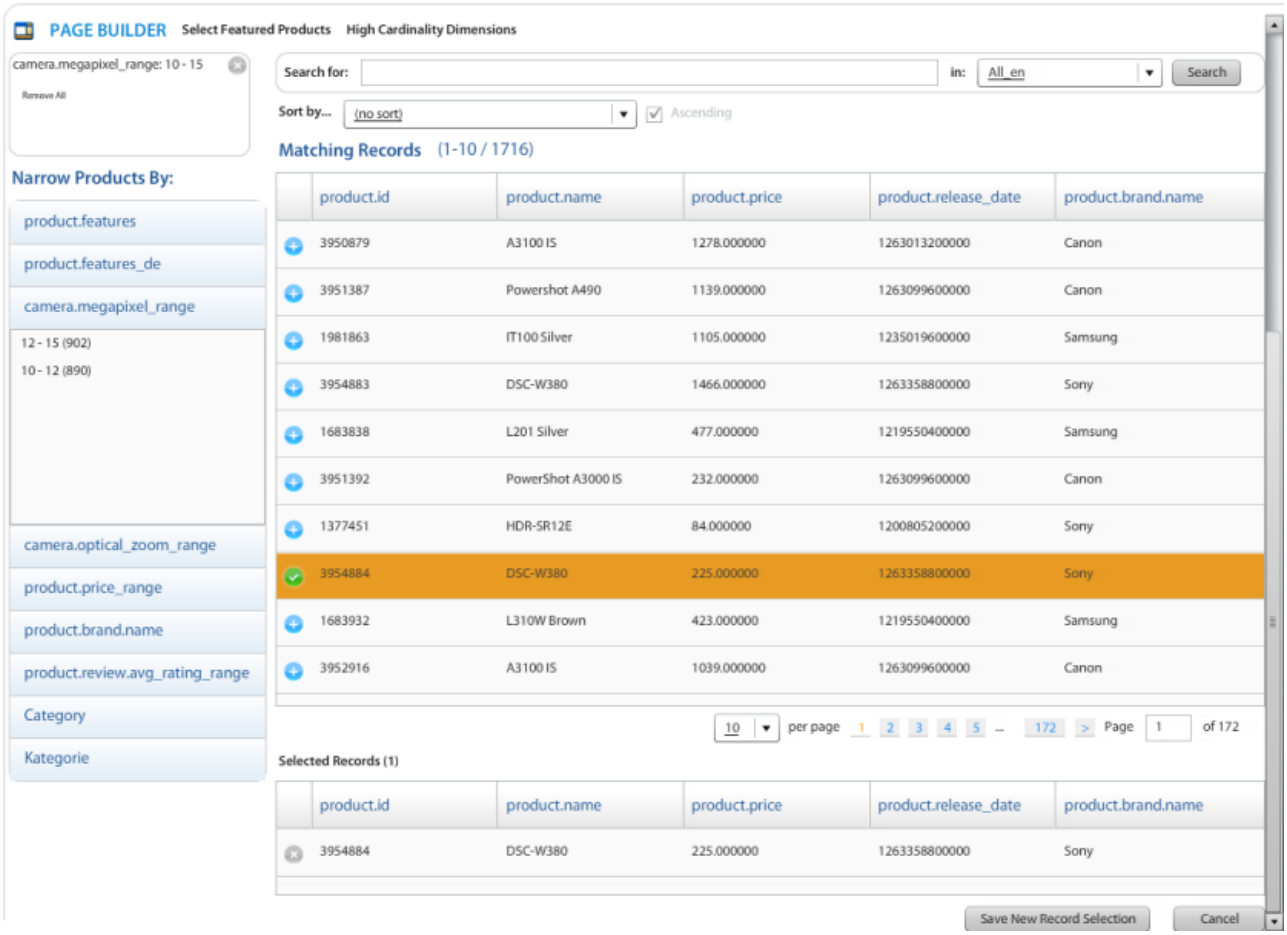
- **Sort by...** dropdown – You can sort results based on an MDEX Engine dimension or property value. The **Ascending** checkbox is selected by default. De-select it to sort in descending order.
- **Filter State** pane – Once selected, refinements and search terms are displayed here. You can click the **X** beside each refinement to remove it individually, or click the **Remove All** button to clear all refinements.

- **Narrow Products By:** pane – You can specify a dynamic page or set of records through navigation refinements instead of or in addition to search results. Selected refinements also display in the **Filter State** pane.
- **Current Matching Records** pane – This pane provides a preview of the records that match the selected filter state, although the exact records that display to the end user may change due to updates in the data set.
- **Pagination Control** – The pagination control allows you to browse matching records page by page. You can also adjust the results per page, or enter a specific page number into the **Page** field to go directly to that results page.
- **Maximum Number of Records** – You can restrict the number of records returned by clicking the radio button next to the combo box and selecting a value between 1 and 20. The **no limit** radio button is enabled by default.
- **Restrict to active refinements** – You can use this checkbox to combine your selected filter state with whatever current filters the end user has already selected. For example, if this editor appears on a Red Wine landing page that triggers on a Wine Type > Red refinement, this checkbox will apply that refinement to the results, whether or not you select it in the editor.

Note: The **in:** drop-down is populated with searchable properties and dimensions from your MDEX Engine. The **Current Matching Records** pane can be configured by modifying your MDEX service configuration file.

Select Featured Products mode

When the microbrowser is used to select specified records rather than a dynamic record result page, the **Sort by...** dropdown and controls appear above the **Matching Records** pane, and the **Maximum Number of Records:** controls are replaced with the **Selected Records** pane. This displays currently selected records, as shown in the screenshot below:



You can add records by clicking the green plus in the leftmost column of the **Matching Records** pane, or remove them by clicking the black X in the leftmost column of the **Selected Records** pane.

Selecting featured records using the microbrowser

The microbrowser enables content administrators to use search and navigation to locate specific records for spotlighting, rather than entering a record id. This functionality is featured in the ThreeRecordBox cartridge.

To select featured records using the microbrowser:

1. In your landing page, add a cartridge that uses the Advanced Record List editor, such as the ThreeRecordBox.
2. Select the **Specific Records** radio button.
3. Click the **Edit List** button.

   In this mode, the microbrowser search and navigation functions are used to find the records you wish to specify. The navigation state itself does not impact which records will be displayed.
4. Search for or navigate to the records you wish to specify and click the green plus button next to each record to add it to the **Selected Records** pane.
5. Click the **Save New Record Selection** button.

The selected records are displayed in the **Content Details Panel** in the editor.

Selecting dynamic records using the microbrowser

The microbrowser enables content administrators to use search and navigation to specify a dynamic page or list of records for spotlighting. This functionality is featured in the ThreeRecordBox cartridge and any cartridge that uses the Link editor.

To select dynamic records using the microbrowser:

1. In your landing page, add a cartridge that uses the Advanced Record List editor, such as the ThreeRecordBox.
2. Select the **Records selected by query** radio button.
3. Click the **Edit Query** button.
4. Use a combination of search queries and navigation refinements to set the desired navigation state.
5. (Optional) Use the **Sort by...** dropdown and **Maximum Number of Records** controls to further configure the sorting behavior and number of records returned.
6. Click the **Save New Filter State** button.

The selected navigation refinements and search terms are displayed in the editor's **Content Details Panel**.

About using aggregated records in the microbrowser

You can configure the microbrowser to display aggregated records instead of regular records by specifying a dimension or property as an aggregation key in your MDEX service definition file.

This allows a merchandiser to group similar records, and spotlight the aggregated records. For example, a promotion can be set to display a selected book, with no differentiation between the hardcover, softcover, and e-book formats.

You can specify your aggregation key in the MDEX Engine service definition file by setting the value of the `<aggregationKey>`element.

Related Links

About the MDEX Engine service definition on page 26

The MDEX Engine service definition file enables you to configure communication information for your MDEX Engine. This information is used by the microbrowser and any editors that contain it.

About the image browser

The image browser enables a content administrator to select images by browsing a content management system. It is a main component of the Image and Image Map editors.

The image browser uses the image service definition file to identify the host and port of the image server that it communicates with.

About the image server service definition

The image server service definition, `images.xml`, identifies the host and port of the server that provides images to the Experience Manager Extensions sample editors.

You can configure it to point Experience Manager at a source of common images, such as a shared repository on a content management system.

Note: The Image editor implementation provided with Experience Manager Extensions is configured specifically to communicate with Sling-based content management systems (a Sling distribution is packaged with the Experience Manager Extensions application). Extending the editor's capabilities to function with other CMS implementations requires custom ActionScript coding and is beyond the scope of this guide.

A default image server service definition is provided out of the box at `<content root>/services/image.xml`. This default definition defines the image root as `<content root>/images` within the same repository that the editors are hosted in. If you edit this configuration, all editors using that configuration file will be updated.

The image server service definition file includes a root element, `<ServiceDefinition>` with the attribute `type="image/sling"` and the following subelements:

Element	Usage
`<title>`	The service title. The Image editor displays this to the user to identify the image repository that is being displayed.
`<host>`	Optional. The hostname or IP address of your image server. By default, the editor attempts to connect to an image server in the same content repository.
`<port>`	Optional. The port that the image server listens on. By default, the editor attempts to connect to an image server in the same content repository.
`<roo¬ tUri>`	The path, relative to the content root (`/`), to the location where the images are stored in the repository.

Following is an example image server service definition:

```
<ServiceDefinition type="image/sling">
    <title>Asset Management System | Spring 2010 | STAGING</title>
        <host>localhost</host>
        <port>8181</port>
        <rootUri>/images</rootUri>
</ServiceDefinition>
```

Note: You can specify a different image server on a per-cartridge basis by creating a new service definition with a new name. By convention, it should live in `<content root>/services` within the JCR repository. To point to the new definition, modify the `imageServiceId` attribute in the editor configuration in the template, as shown below. This applies to any type of editor that uses the image browser.

```
  <editors:ImageEditor propertyName="image" label="Image"
    imageServiceId="/services/image2"/>
```

The `imageServiceId` attribute is the path to the service definition file, relative to the context root, omitting the `.xml` extension.

Related Links

Updating the service definitions and zone configuration file on page 37
the Experience Manager Extensions sample editors use service definition files that define how to communicate with external resources, such as an MDEX Engine or an image server.

Uploading images to the content repository

The image browser enables content administrators to select images by browsing a hierarchical image repository. Experience Manager Extensions includes such a repository that can be used to serve images.

The repository comes with several sample images by default. You can access the repository via WebDAV to upload additional images.

To upload images to the repository:

1. Establish a WebDAV connection to the content repository.

   In a default installation within the Endeca Tools Service, this located at a URL like `http://localhost:8006/oexmx`.
2. Navigate to the image root directory.

   By convention, the editors look for images in the `image` directory, but this can be specified as part of the editor definition in a Experience Manager template.
3. Create any subdirectories as needed for organization.
4. Upload your images to this location.

Related Links

Prerequisites on page 9
This section describes the software requirements for installing and using Experience Manager Extensions.

Updating the service definitions and zone configuration file

the Experience Manager Extensions sample editors use service definition files that define how to communicate with external resources, such as an MDEX Engine or an image server.

The following sample files are provided:

File	Description
`images.xml`	Specifies configuration for connecting to a content management system that contains images. This configuration is used by the Image editor and Image Map editor.
`mdex.xml`	Specifies configuration for connecting to an MDEX Engine. This configuration is used by the microbrowser and any editors that contain it.
`zones.xml`	Specifies configuration for zones that are used to contain dynamic slot content. This configuration is used by the Dynamic Slot editor.

These files are located in the JCR repository under `<content root>/services` and can be updated via WebDAV.

Related Links

About the MDEX Engine service definition on page 26

The MDEX Engine service definition file enables you to configure communication information for your MDEX Engine. This information is used by the microbrowser and any editors that contain it.

About the image server service definition on page 35

The image server service definition, `images.xml`, identifies the host and port of the server that provides images to the Experience Manager Extensions sample editors.

About the zone configuration file for dynamic slots on page 54

The zone configuration file for Experience Manager specifies which rule zones can be used for dynamic slot content, as well as what content type they can contain (for example, `SideBarItem` or `FullWidthContent` cartridges).

Chapter 3

Working with Advanced Feature Cartridges

Experience Manager Extensions includes components that support new MDEX Engine features such as record boost and bury, dimension value boost and bury, and advanced dimension search functionality.

About boost and bury

Beginning with MDEX Engine 6.1.4, business users have more control over the presentation of dimension values, as well as enhanced control over the relevancy ranking and sort order of search results.

MDEX Engine 6.1.4 includes a new module called stratification, which can be used to group dimension values or records, and then use those groups as part of ordering them for display. While the MDEX Engine contains API support to utilize these modules, there is no out-of-the-box support for business users to configure boost and bury within Workbench. Experience Manager Extensions contains a set of assets to enable these new features for Experience Manager-powered applications.

Record boost and bury can be used to promote certain specific records (product_id = 12345 or product_id = 23423) or records based off of a navigation query (price greater than $100 or the Brand = Acme). Unlike traditional Endeca business rules, these promoted products would be boosted to the top (or the bottom) of the actual search results. This gives the merchandising team unprecedented control over the ordering and display of products.

Dimension value boost and bury can be used to control the order of dimension values in the Guided Navigation. Currently, dimension value ordering is controlled by configuration in Developer Studio, or by using the Dimension Ordering component of Workbench. However, that configuration is global across the entire site. Dimension value boost and bury allows for the merchandising team to control the ordering of dimensions on a query-by-query basis.

The assets presented in Experience Manager Extensions are meant to serve as reference implementations. It is likely that application-specific modifications will be necessary in order to fulfill that application's requirements. The source code for both the Flex editors and the tag handlers are included.

Related Links

Overview of implementing record boost and bury on page 40
Using the included assets to implement record boost and bury involves creating a new cartridge template (replacing the existing Results List template) and using the new cartridge in pages configured in Experience Manager.

Overview of implementing dimension value boost and bury on page 42

Using the included assets to implement dimension value boost and bury involves the creation of a new cartridge template (replacing the existing Guided Navigation template), and using the new cartridge in pages configured in Experience Manager.

Overview of implementing record boost and bury

Using the included assets to implement record boost and bury involves creating a new cartridge template (replacing the existing Results List template) and using the new cartridge in pages configured in Experience Manager.

There are two main configuration steps for records value boost and bury:

- **Creating the cartridge template:** This new XML template replaces the existing Results List cartridge.
- **Updating the template mapping in the application:** This new cartridge then must be mapped to the corresponding UI controls, instead of the existing Results List cartridge.

Experience Manager Extensions includes sample cartridge templates for both the existing Results List cartridge, as well as one that includes the new record boost and bury configuration. After processing, they both return a list of records in the same format, so either one can be used in the application and mapped to the same UI rendering elements for display.

Creating cartridge templates for record boost and bury

Record boost and bury requires a different cartridge template configuration than the existing Results List cartridge.

This cartridge is a replacement for the existing Results List cartridge. It uses an unconventional structure so as to utilize other Experience Manager Extension Flex editors (such as the microbrowser) without creating a completely separate editor.

The best approach is to take the sample cartridge located at `\pbx_wine_app\page_builder_templates\MainColumnContent-ResultsListBoost¬Bury.xml` and make slight modifications.

To create a cartridge template for record boost and bury:

1. Copy the sample Results List cartridge XML:
2. Modify the `<recordSpec>` definition, the `<recsPerPage>` definition in the Property definition section.

   The following is a complete list of the Property elements found in the element. Some can be modified, most cannot.

Attribute	Description
`boostStrata` and `buryStrata`	These are internal variables used by the Flex editor to configure the products to be boosted/buried.
`sort`	This is an internal variable used to control overriding the sorting of products.
`relrank`	This is an internal variable used to control overriding the relevancy ranking of products.
`recordSpec`	This value is the property found on the records that determines uniqueness when boosting and burying speicifc products. If your application uses aggregate records, this value should be the rollup key.

Attribute	Description
	If you're not using aggregate records, this should be the record spec property as configured in Developer Studio.
recsPerPage	The default value of the records per page. You should configure this for your site's default value.
navigation_records	Internal variable used to trigger the xtag handler for boosting and burying. This property should be the last one specified in the cartridge XML.

3. Within the editor configuration, enter the sort keys, relevancy ranking module changes and records per page choices.

An example of a complete template is reproduced below:

```
<ContentTemplate xmlns="http://endeca.com/schema/content-template/2008"
    xmlns:editors="editors"
    xmlns:xavia="http://endeca.com/schema/xavia/2010"
    type="MainColumnContent"
    id="ResultsList">

  <Description>Displays search and navigation results in a list or grid
view.</Description>
  <ThumbnailUrl>/thumbnails/ResultsList.png</ThumbnailUrl>

  <ContentItem>
    <Name>New Results List</Name>
    <Property name="boostStrata">
        <xavia:List/>
    </Property>
    <Property name="buryStrata">
        <xavia:List/>
    </Property>
    <Property name="sort">
        <xavia:Item>
            <Property name="label">Default</Property>
            <Property name="attribute"/>
            <Property name="ascending"/>
        </xavia:Item>
    </Property>
    <Property name="relrank">
        <String/>
    </Property>
    <Property name="recordSpec">
        <String>P_WineID</String>
    </Property>
    <Property name="recsPerPage">
        <String>10</String>
    </Property>
 <!-- The BoostBuryResultsProcessor MUST be last in the definition of
properties -->
 <Property name="navigation_records">
  <BoostBuryResultsProcessor xmlns="http://endeca.com/schema/con¬
tent/xtags/2010"/>
 </Property>
  </ContentItem>

  <EditorPanel>
    <BasicContentItemEditor>
```

```
      <editors:SortEditor propertyName="sort" label="Sorts">
          <choice label="Default" />
          <choice label="Score (Descending)" value="P_Score" />
          <choice label="Price (Ascending)" value="P_Price" />
          <choice label="Price (Descending)" value="P_Price" />
      </editors:SortEditor>
     <editors:ChoiceEditor propertyName="relrank" label="Relevance ranking">

          <choice label="Default" />
          <choice label="First" value="first" />
          <choice label="By Price (Static)" value="static(P_Price)" />
          <choice label="Frequency" value="freq" />
      </editors:ChoiceEditor>
      <editors:ChoiceEditor propertyName="recsPerPage" label="Records per
 page">
                <choice label="10" value="10"/>
                <choice label="20" value="20"/>
                <choice label="30" value="30"/>
                <choice label="50" value="50"/>
                <choice label="100" value="100"/>
            </editors:ChoiceEditor>
      <editors:RecordStratificationEditor propertyName="boostStrata"
buryProperty="buryStrata" label="Customize Results List" />
    </BasicContentItemEditor>
  </EditorPanel>
</ContentTemplate>
```

Modifying the application to display the records

The new boost and bury cartridge is meant to serve as a complete replacement for the existing Results List cartridge. As such, it returns results in the exact same format as the existing cartridge.

To support the new cartridge in your application, you should only need to update the mapping mechanism which determines which JSP or ASP page renders each Experience Manager cartridge. If your application already uses the search results cartridge, the mapping for the existing cartridge should apply to the new cartridge unless you change the ID of one of the cartridges.

In the included `PbxRefApp` for Java, the `templateconfig.properties` found under `\PbxRefApp\WEB-INF\classes` contains the following:

```
ResultsList=/cartridges/ResultsList.jsp
ResultsListOriginal=/cartridges/ResultsList.jsp
```

In the included `PbxRefAppDotNet` for the RAD Toolkit for ASP.NET, the `Web.config` contains the following:

```
<add templateId="ResultsList" handlerPath="~/Resources/ContentCon¬
trols/Records/ResultsList.ascx" />
<add templateId="ResultsListOriginal" handlerPath="~/Resources/ContentCon¬
trols/Records/ResultsList.ascx" />
```

Overview of implementing dimension value boost and bury

Using the included assets to implement dimension value boost and bury involves the creation of a new cartridge template (replacing the existing Guided Navigation template), and using the new cartridge in pages configured in Experience Manager.

There are two main configuration steps for dimension value boost and bury:

- **Creating the cartridge template:** This new XML template replaces the existing Guided Navigation cartridge.
- **Updating the template mapping in the application:** This new cartridge then must be mapped to the corresponding UI controls, instead of the existing Guided Navigation cartridge.

Experience Manager Extensions includes sample cartridge templates for both the existing Guided Navigation cartridge, as well as the new dimension value boost and bury configuration. After processing, they both return a list of dimensions in the same format, so either one can be used in the application, and mapped to the same UI rendering elements for display.

The dimension value boost and bury Flex editor is a very basic editor, serving as a reference implementation primarily for the boost and bury feature.

Creating cartridge templates for dimension value boost and bury

Dimension value boost and bury requires a different cartridge template configuration than the existing Guided Navigation cartridge.

This cartridge is a replacement for the existing Guided Navigation cartridge. It uses a different namespace for the Content Item property, and uses a new Flex editor for configuration.

To create a cartridge template for dimension value boost and bury:

1. Create a new cartridge template. For example:

```
<ContentTemplate xmlns="http://endeca.com/schema/content-template/2008"

xmlns:xtags="http://endeca.com/schema/content/xtags/2010"
type="SidebarItem"
id="GuidedNavigationBoostBury"
 xmlns:editors="editors">
  <Description>Displays Endeca Guided Navigation with boost/bury op¬
tions.</Description>
  <ThumbnailUrl>/thumbnails/PageTemplate/GuidedNavigation.png</Thumbnail¬
Url>
  <ContentItem>
  </ContentItem>
  <EditorPanel>
  </EditorPanel>
</ContentTemplate>
```

2. Within the `<ContentItem>` definition, specify a `<NavigationRefinementConfig>` property. For example:

```
<ContentItem>
    <Name>New Guided Navigation</Name>
    <Property name="refinements">
      <NavigationRefinementConfig xmlns="http://endeca.com/schema/con¬
tent/xtags/2010" />
    </Property>
 <Property name="body">
      <String/>
    </Property>
  </ContentItem>
```

3. Within the <EditorPanel> definition, specify a <CABoostBuryEditor> that is bound to the <NavigationRefinementConfig> property. For example:

```
<EditorPanel>
    <BasicContentItemEditor>
     <editors:CABoostBuryEditor propertyName="refinements" label="refine¬
ments" enabled="true"/>
   <StringEditor propertyName="body" label="Body" enabled="true"
height="108"/>
    </BasicContentItemEditor>
  </EditorPanel>
```

An example of a complete template is reproduced below:

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentTemplate xmlns="http://endeca.com/schema/content-template/2008"
xmlns:xtags="http://endeca.com/schema/content/xtags/2010"
type="SidebarItem"
id="GuidedNavigationBoostBury"
 xmlns:editors="editors">
  <Description>Displays Endeca Guided Navigation with boost/bury options.</De¬
scription>
  <ThumbnailUrl>/thumbnails/PageTemplate/GuidedNavigation.png</ThumbnailUrl>

  <ContentItem>
    <Name>New Guided Navigation</Name>
    <Property name="refinements">
      <RefinementDimensionConfigs xmlns="http://endeca.com/schema/con¬
tent/xtags/2010">
      </RefinementDimensionConfigs>
    </Property>
    <Property name="body">
      <String/>
    </Property>
  </ContentItem>
  <EditorPanel>
    <BasicContentItemEditor>
      <editors:CABoostBuryEditor propertyName="refinements" label="refine¬
ments" enabled="true"/>
   <StringEditor propertyName="body" label="Body" enabled="true"
height="108"/>
    </BasicContentItemEditor>
  </EditorPanel>
</ContentTemplate>
```

Modifying the application to display the dimension values

If your application already uses the existing Guided Navigation cartridge, only slight modifications are necessary to use the Boost Bury configured Guided Navigation cartridge.

The new Boost and Bury cartridge returns a dimension list object in the exact same format as the existing Guided Naviagtion cartridge. As such, you should only need to update the mapping mechanism which determines which JSP or ASP page renders each Experience Manager cartridge. This mapping should create a mapping for both the existing Guided Navigation cartridge, as well as the ID for the new cartridge.

In the included `PbxRefApp` for Java, the `templateconfig.properties` found under `\PbxRefApp\WEB-INF\classes` contains the following:

```
GuidedNavigation=/cartridges/GuidedNavigation.jsp
GuidedNavigationBoostBury=/cartridges/GuidedNavigation.jsp
```

In the included `PbxRefAppDotNet` for the RAD Toolkit for ASP.NET, the `Web.config` contains the following:

```
<add templateId="GuidedNavigation" handlerPath="~/Resources/ContentCon¬
trols/Navigation/GuidedNavigation.ascx" />
 <add templateId="GuidedNavigationBoostBury" handlerPath="~/Resources/Con¬
tentControls/Navigation/GuidedNavigation.ascx" />
```

About the advanced dimension search features

Beginning with MDEX Engine 6.2.0, business users can have the ability to restrict dimension search to a specific set of dimensions, and to display refinement counts in dimension search.

Experience Manager Extensions contains a set of assets to enable these new features for Experience Manager-powered applications, including a new editor to select a list of dimensions for dimension search and tag handlers for Java and .NET.

The assets presented in Experience Manager Extensions are meant to serve as reference implementations. It is likely that application-specific modifications will be necessary in order to fulfill that application's requirements. The source code for both the Flex editor and the tag handlers are included.

Overview of implementing enhanced dimension search

Using the included assets to implement the enhanced dimension search cartridge involves creating a new cartridge template (replacing the existing Dimension Search Results template) and using the new cartridge in pages configured in Experience Manager.

There are two main configuration steps for records value boost and bury:

- **Creating the cartridge template:** This new XML template replaces the existing Dimension Search Results cartridge.
- **Updating the template mapping in the application:** This new cartridge then must be mapped to the corresponding UI controls, instead of the existing Dimension Search Results cartridge.

Experience Manager Extensions includes an updated sample template for Dimension Search Results that enables the new functionality.

Creating the cartridge template for enhanced dimension search

Enhanced dimension search requires a different cartridge template configuration than the existing Dimension Search Results cartridge.

This cartridge is a replacement for the existing Dimension Search Results cartridge, with the addition of a new `<DimensionSearchConfig>` property and an editor that enables configuration of dimension search results.

To create a cartridge template for record boost and bury:

1. Create a new cartridge template. For example:

```
<ContentTemplate xmlns="http://endeca.com/schema/content-template/2008"
  xmlns:editors="editors"
  xmlns:xtags="http://endeca.com/schema/content/xtags/2010"
  xmlns:xavia="http://endeca.com/schema/xavia/2010"
  type="MainColumnContent" id="DimensionSearchResults">
  <Description>Displays dimension search results.</Description>
  <ThumbnailUrl>/thumbnails/PageTemplate/DimensionSearchResults.png</Thumb¬
nailUrl>
  <ContentItem>
  </ContentItem>
  <EditorPanel>
  </EditorPanel>
</ContentTemplate>
```

2. Within the <ContentItem> definition, specify a <NavigationRefinementConfig> property. For example:

```
  <ContentItem>
    <Name>New Dimension Search Results</Name>
    <Property name="dimSearchConfig">
        <xtags:DimensionSearchConfig>
            <xavia:Item>
                <xavia:Property name="dimensionList">
                  <xavia:List/>
                </xavia:Property>
                <xavia:Property name="showCount">true</xavia:Property>
            </xavia:Item>
        </xtags:DimensionSearchConfig>
    </Property>
  </ContentItem>
```

3. Within the <EditorPanel> definition, specify a <CABoostBuryEditor> that is bound to the <NavigationRefinementConfig> property. For example:

```
  <EditorPanel>
    <BasicContentItemEditor>
      <editors:DimensionSearchEditor propertyName="dimSearchConfig" la¬
bel="Dimensions To Search"/>
    </BasicContentItemEditor>
  </EditorPanel>
```

An example of a complete template is reproduced below:

```
<ContentTemplate xmlns="http://endeca.com/schema/content-template/2008"
  xmlns:editors="editors"
  xmlns:xtags="http://endeca.com/schema/content/xtags/2010"
  xmlns:xavia="http://endeca.com/schema/xavia/2010"
  type="MainColumnContent" id="DimensionSearchResults">
  <Description>Displays dimension search results.</Description>
  <ThumbnailUrl>/thumbnails/PageTemplate/DimensionSearchResults.png</Thumb¬
nailUrl>
  <ContentItem>
    <Name>New Dimension Search Results</Name>
    <Property name="dimSearchConfig">
        <xtags:DimensionSearchConfig>
            <xavia:Item>
                <xavia:Property name="dimensionList">
                  <xavia:List/>
                </xavia:Property>
```

```
                <xavia:Property name="showCount">true</xavia:Property>
            </xavia:Item>
        </xtags:DimensionSearchConfig>
    </Property>
  </ContentItem>

  <EditorPanel>
    <BasicContentItemEditor>
      <editors:DimensionSearchEditor propertyName="dimSearchConfig" la¬
bel="Dimensions To Search"/>
    </BasicContentItemEditor>
  </EditorPanel>
</ContentTemplate>
```

Modifying the application for enhanced dimension search

The enhanced dimension search cartridge is meant to serve as a replacement for the existing Dimension Search Results cartridge.

To support the new cartridge in your application, you should only need to update the mapping mechanism which determines which JSP or ASP page renders each Experience Manager cartridge. If your application already uses the search results cartridge, the mapping for the existing cartridge should apply to the new cartridge unless you change the ID of one of the cartridges.

In the included `PbxRefApp` for Java, the `templateconfig.properties` found under `\PbxRefApp\WEB-INF\classes` contains the following:

```
DimensionSearchResults=/cartridges/DimensionSearchResults.jsp
```

In the included `PbxRefAppDotNet` for the RAD Toolkit for ASP.NET, the `Web.config` contains the following:

```
<add templateId="DimensionSearchResults" handlerPath="~/Resources/Content¬
Controls/Search/DimensionSearchResults.ascx" />
```

Chapter 4

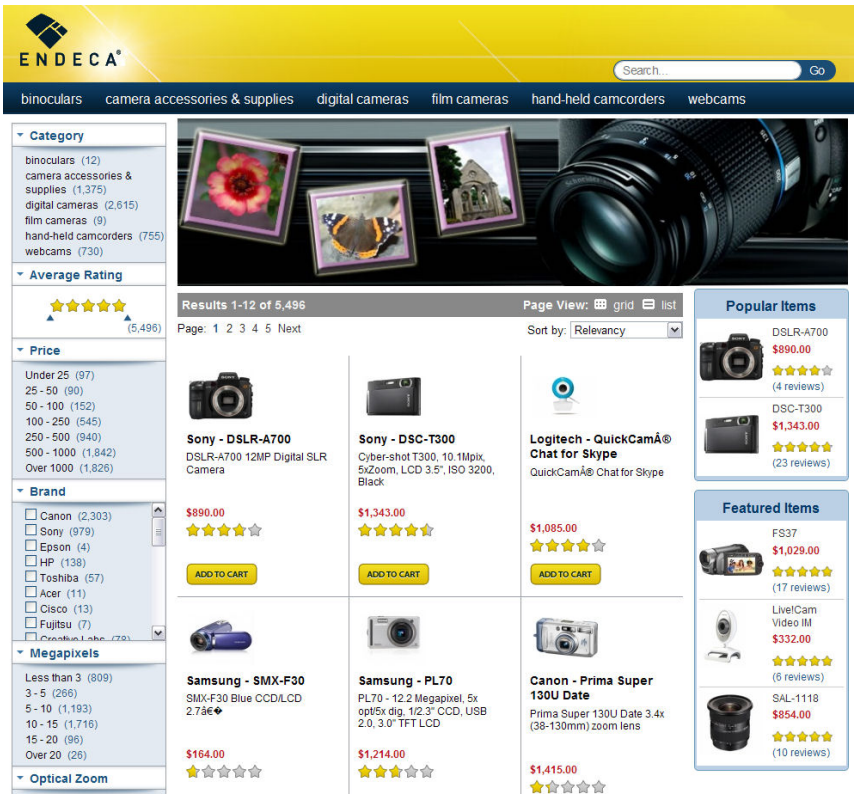
Working with Dynamic Slots

Experience Manager Extensions includes components that support dynamic slots functionality, enabling more granular content management within pages created in Experience Manager.

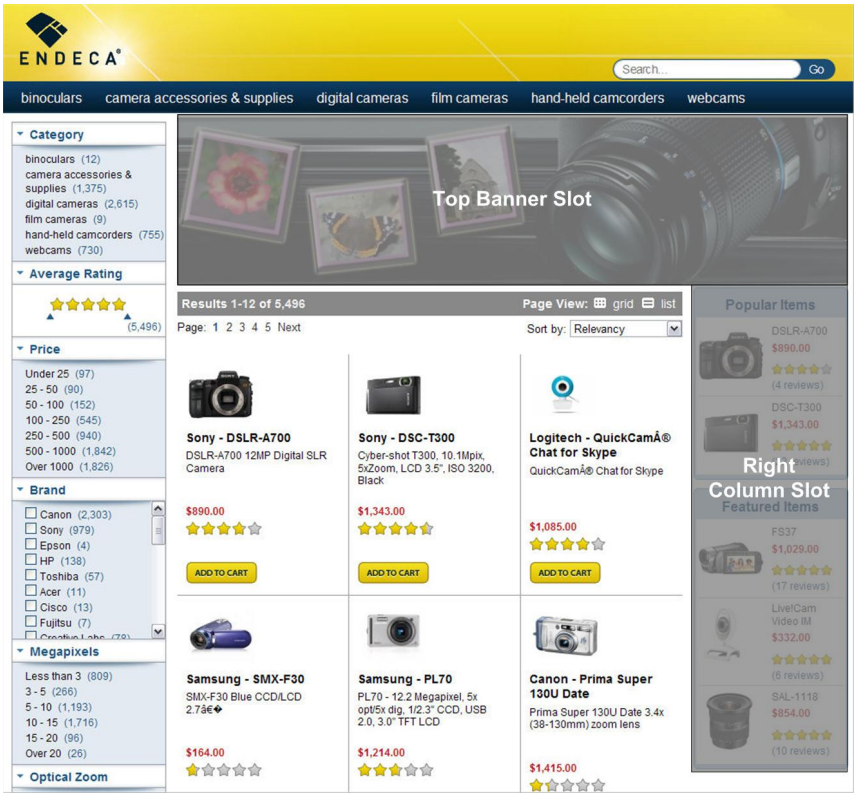
About dynamic slots

The Dynamic Slot editor combined with the Dynamic Slot tag handler enable content administrators to manage the content for specific sections of a Experience Manager-driven page independently from the overall page.

For example, you can specify the content to display in a particular section based on different trigger criteria from the page trigger, or give control over certain page sections to a different set of users from the ones who are responsible for the page as a whole. Consider a page such as the following:



Typically, all the elements on a page are managed as a single "landing page" within Experience Manager. Each component on the page (such as a menu, search results grid, image, or promotion) is an individual cartridge on the page. However, in some cases it may make sense to use a single landing page to drive the majority of the page (including the main menu, Guided Navigation, and search results), but manage the banner image and the right sidebar content independently.



In this case, the image that displays in the Top Banner area is not specified directly in the configuration for this page. Instead, a content administrator inserts a placeholder for the "slot" and then specifies the content that populates the slot separately. As end users browse the site, the top banner slot is populated dynamically with the content that is returned for that slot. As you can see in the Right Column area, this placeholder slot can contain more than one content item or cartridge. The configuration for each of the remaining page sections is contained in the landing page itself.

Dynamic slots can address the following use cases:

- A brand manager needs to control the banner images that display throughout the site. This is a different person from the merchandiser who typically manages pages in Experience Manager.
- A brand manager needs to be able to specify the images that display at a particular navigation state (for example, Digital Cameras > Samsung) even if there is no specific landing page for that navigation state.
- A merchandiser wishes to display promotions in the sidebar area based on more specific trigger criteria than those that apply to the page as a whole. For example, one could create a page to use as a base for all "Digital Cameras" pages, and populate the sidebar sections with more specific content based on the brand, price range, or other dimensions. This model enables content reuse for most of the content within a page with page-specific overrides for subsections as needed. It removes the need to create many individual pages for each specific combination of triggers.
- A merchandiser wishes to display promotions in the sidebar area based on trigger criteria that are simply different from those on the page as a whole. For example, there might be a "Back to School" special for a particular time frame that applies to all pages within a category or even the entire site.

This model enables content reuse for individual sections across a variety of pages. The reusable sections are managed in a central location so that updates immediately take effect across all the pages that include the reused content, rather than having to edit each one manually.

Overview of implementing dynamic slots

Using the dynamic slots feature involves both developer setup and content administrator setup that is distinct from the everyday maintenance activities of managing dynamic slot content.

There are two main aspects of the dynamic slot feature that work in concert:

- **Dynamic slot cartridges** create placeholders on a specific landing page for dynamic content that is populated at query time.
- **Dynamic slot content** represents individual cartridges that are managed independently of landing pages that populate the dynamic slots. Due to a limitation of the Experience Manager tool, dynamic slot content is managed as individual "landing pages" in Experience Manager.

Implementing dynamic slots involves the following high-level tasks:

Developer setup

To enable an application for dynamic slots:

- The application's instance configuration must have rule zones and groups configured for use with dynamic slots
- There must be a zone configuration file present in the Experience Manager Extensions repository.
- There must be templates designed for dynamic slot cartridges that are placed on a page, as well as for standalone pieces of dynamic slot content.
- There must be rendering code in the front-end application that can handle the dynamic slot content.

Experience Manager Extensions includes a Deployment Template module that deploys a version of the sample wine data with the proper instance configuration for dynamic slots. It also includes a sample zone configuration file, Experience Manager templates, and rendering code for dynamic slots.

Content administrator setup

With a dynamic slot–enabled application, a content adminstrator can add dynamic slot cartridges to landing pages in Experience Manager. This creates placeholders for dynamic slot content, which is maintained separately with their own triggering criteria. After setting up dynamic slot placeholders on the appropriate pages, the dynamic slot cartridges generally do not need to be updated except to add or remove them, or to move them to different sections of the page.

Content administrator ongoing content management

Once dynamic slot cartridges have been added to landing pages in the application, a content administrator can manage the content that populates these slots by creating and configuring standalone pieces of dynamic slot content. By assigning the proper triggers and priority to these pieces of dynamic slot content, content administrators can manage the content across a wide range of pages with relatively little manual effort.

As a best practice, content that is relatively stable should be managed as part of the landing page configuration, while content that changes more frequently should be managed as dynamic slot content.

Related Links

Developer configuration for dynamic slots on page 52

Enabling dynamic slots requires more supporting configuration than regular landing pages in Experience Manager.

Working with dynamic slots in Experience Manager on page 59
Using the dynamic slots feature within Experience Manager is a two-part process: adding a placeholder for dynamic slot content on a page, and creating the pieces of dynamic slot content themselves.

Developer configuration for dynamic slots

Enabling dynamic slots requires more supporting configuration than regular landing pages in Experience Manager.

Before content administrators can use the dynamic slots feature, you must create additional rule zones, rule groups, and a configuration file for Experience Manager.

In addition to the above configuration, you must also create templates for both dynamic slot cartridges and for standalone dynamic slot content, and write rendering code for your front-end application that can handle the dynamic slot content.

Experience Manager Extensions includes a sample wine application that is enabled for dynamic slots by default, including all application and tools configuration, Experience Manager templates, and rendering code. The following components are provided with Experience Manager Extensions installation:

Rule zones	The following zones are set up for dynamic slots in the enhanced sample wine application: • CenterColumnZone • LeftColumnZone • RightColumnZone • HeaderZone • FooterZone
Rule groups	The following rule groups are set up for dynamic slots in the enhanced sample wine application: • CenterColumnGroup • LeftColumnGroup • RightColumnGroup • HeaderGroup • FooterGroup
Zone configuration file	A sample zone configuration file is provided. It is stored in the JCR repository at `http://workbench-host:8006/pbx/services/zones.xml`.
Dynamic slot cartridge templates	These templates are used to create a placeholder on a landing page for dynamic slot content. The following templates are provided in `APP_DIR/config/page_builder_templates` within your deployed application: • FullWidthContent-FullWidthContentSlot.xml • MainColumnContent-MainColumnContentSlot.xml

	• SidebarItem-SidebarItemSlot.xml
Dynamic slot content templates	These templates are used as a basis for creating the standalone pieces of dynamic slot content. The following templates are provided in *APP_DIR*/config/page_builder_templates within your deployed application: • PageTemplate-01-HeaderZoneWidget.xml • PageTemplate-03-CenterColumnZoneWidget.xml • PageTemplate-05-FooterZoneWidget.xml • PageTemplate-02-LeftColumnZoneWidget.xml • PageTemplate-04-RightColumnZoneWidget.xml
Rendering code	Rendering code is provided and configured to handle content created using any of the dynamic slot cartridge templates listed above: • For Java, the renderer is located in /reference/Con¬tentAssemblerRefApp/cartridges/DynamicSlot.jsp. • For the RAD Toolkit for .NET, the renderer is located in refer¬ence\PbxRefAppDotNet\Resources\ContentCon¬trols\Slots.

Related Links

Deploying the enhanced sample wine application on page 15
To configure the sample wine application, run the Deployment Template deploy script using the default options and use the included control scripts to initialize your application.

Deploying the Experience Manager Extensions reference application for Java on page 16
You can install the Experience Manager Extensions reference application into the Endeca Tools Service by using the sample context file included in the installation package.

Deploying the Experience Manager Extensions reference application for the RAD Toolkit for ASP.NET on page 16
You can install the Experience Manager Extensions reference application into an instance of IIS 6 or 7.

Creating rule zones for dynamic slots

In addition to the rule zones in your application for landing pages, you should create additional zones for each section of the page that you want to enable for dynamic slots.

Recall that landing pages use the dynamic business rule mechanism for triggering. A zone is a logical grouping of rules whose triggers are evaluated against each other to determine which ones are returned for a particular query. Creating additional zones for dynamic slots (as opposed to pages) enables content administrators to specify different triggering criteria for the dynamic slot content than for the page as a whole. Creating separate zones for each section of the page that can have dynamic slot content enables the triggers for each section to be evaluated independently of each other.

The procedure to create a zone for dynamic slots is the same as for pages, with the exception that the rule limit can be greater than 1. This enables a slot to contain multiple dynamic content items or cartridges.

To create zones for dynamic slots:

1. In the **Project Explorer** of Developer Studio, expand **Dynamic Business Rules**.

2. Double-click **Zones** to open the **Zones** view.
3. Click **New** to open the **New Zone** editor.
4. In the **Name** field, provide a unique name for the zone, such as `LeftColumnSlot`.
5. In the **Rule Limit** field, enter a number greater than or equal to `1`.

   This is the maximum number of dynamic slot content items that can be returned for this section of the page.
6. Select **Valid for search**.

   Leave **Shuffle rules** and **Unique by this dimension/property** unselected.
7. Click **OK**.

The dynamic slots feature does not require the configuration of additional rule styles. For information about best practices for rule styles with Experience Manager, see the *Oracle Endeca Experience Manager Developer's Guide*.

Creating rule groups for dynamic slots

As a best practice, you should create a rule group for each rule zone that you have set up for dynamic slot content.

Separate rule groups enable content administrators to organize the reusable dynamic slot content within the Experience Manager interface. They also enable a Workbench administrator to control which users have permissions to manage the content in pages or dynamic slots.

Note: Experience Manager provides no mechanism to enforce what kind of pages or dynamic slot items are created within any rule group. The association between a rule zone and a rule group is entirely a matter of convention. For this reason, Oracle recommends that you give rule groups that are meant to be used for dynamic slot content a name that matches the associated rule zone, and that this name be meaningful to a content administrator using the tool.

To create rule groups for dynamic slots:

1. In the **Project Explorer** of Developer Studio, expand **Dynamic Business Rules**.
2. Double-click **Rules** to open the **Rules** view.
3. In the **Rule groups** pane on the left, select the name of any existing rule group such as **default** and click **New** to open the **New Rule Group** editor.
4. In the **Group name** field, provide a unique name for the group, such as `LeftColumnSlot`.

   Do not add any rules to the new group.
5. Click **OK**.

About the zone configuration file for dynamic slots

The zone configuration file for Experience Manager specifies which rule zones can be used for dynamic slot content, as well as what content type they can contain (for example, `SidebarItem` or `FullWidth¬Content` cartridges).

The zone configuration file is used by the Dynamic Slots editor the same way that other editors use the service definitions for the MDEX Engine and image server. By convention, the configuration file is named `zones.xml`, and its contents should be similar to the following:

```
<?xml version="1.0" encoding="UTF-8"?>

<Zones>
  <Zone name="HeaderSlot" type="FullWidthContent" maxItems="3" />
  <Zone name="LeftColumnSlot" type="SidebarItem" maxItems="5" />
  <Zone name="CenterColumnSlot" type="MainColumnContent" maxItems="5" />
  <Zone name="RightColumnSlot" type="SidebarItem" maxItems="5" />
  <Zone name="FooterSlot" type="FullWidthContent" maxItems="3" />
</Zones>
```

For each rule zone you have set up for dynamic slot content, you must specify a `<Zone>` element with the following attributes:

Attribute	Description
`name`	The name of the zone as defined in your application's instance configuration.
`type`	The type of dynamic slot content items that should be created in this zone. This controls which zones are available for a particular cartridge type in the Dynamic Slot editor in Experience Manager.
`maxItems`	The maximum number of content items that can be returned for this zone. This number should be the same as the **Rule limit** specified in for the zone in Developer Studio. A content administrator can further limit this number on a per-page basis.

Related Links

Updating the service definitions and zone configuration file on page 37
the Experience Manager Extensions sample editors use service definition files that define how to communicate with external resources, such as an MDEX Engine or an image server.

Creating templates for dynamic slot cartridges

The cartridge template adds a dynamic slot to a landing page. This is a placeholder for dynamic slot content that is populated at query time.

Dynamic slot cartridges work the same way as regular cartridges, except that instead of configuring the content directly in the cartridge itself, a content administrator specifies a zone from which to fetch dynamic slot content when the page is triggered and a maximum number of content items to display in that slot.

As a best practice, you should create one dynamic slot cartridge template for each content item type you want to enable for dynamic slots, such as `SidebarItem` or `MainColumnContent`.

To create a cartridge template for adding dynamic slots:

1. Create a new cartridge template. For example:

```
<ContentTemplate xmlns="http://endeca.com/schema/content-template/2008"
  xmlns:editors="editors" type="SidebarItem" id="SidebarSlot">
```

```
  <Description>A placeholder for dynamic slot content.</Description>
</ContentTemplate>
```

As with any cartridge template, the `type` attribute controls the kind of section in which this cartridge can be inserted. In this example, the cartridge is intended for content to be displayed in a sidebar column.

2. Within the `<ContentItem>` definition, specify a dynamic slot property. For example:

```
  <ContentItem>
    <Name>Dynamic slot</Name>
    <Property name="contents">
      <DynamicSlot xmlns="http://endeca.com/schema/content/xtags/2010"
        type="SidebarItem" zone="LeftColumnSlot" maxContentItems="3"/>
    </Property>
  </ContentItem>
```

The property name can be any string you wish. The attributes of the `<DynamicSlot>` element are as follows:

Attribute	Description
`xmlns`	The namespace of the `<DynamicSlot>` element. The value must be `http://endeca.com/schema/content/xtags/2010` in order to enable the Dynamic Slot tag handler for the Content Assembler API.
`type`	The type of content items that this slot can contain. As a best practice, this should be the same as the type of the cartridge template. This value is not configurable in Experience Manager.
`zone`	The default zone from which to fetch dynamic slot content for this cartridge. This can be configured by a content administrator. The available choices are determined by the zones specified in the zone configuration file that are associated in with the content item `type` for the Dynamic Slot editor.
`maxCon¬ tentItems`	The default value for the maximum number of dynamic slot items to return in this section on this page. A content administrator can change this value, up to the limit specified for this zone in the zone configuration file.

3. Within the `<EditorPanel>` definition, specify a Dynamic Slot editor that is bound to the dynamic slot property. For example:

```
  <EditorPanel>
    <BasicContentItemEditor>
      <editors:DynamicSlotEditor propertyName="contents"
        zoneConfig="services/zones"/>
    </BasicContentItemEditor>
  </EditorPanel>
```

The `zoneConfig` attribute specifies the location of the zone configuration file, relative to the context root, omitting the `.xml` extension.

An example of a complete template is reproduced below:

```
<ContentTemplate xmlns="http://endeca.com/schema/content-template/2008"
  xmlns:editors="editors" type="SidebarItem" id="SidebarSlot">
  <Description>A placeholder for dynamic slot content.</Description>
  <ContentItem>
    <Name>Dynamic slot content</Name>
    <Property name="contents">
      <DynamicSlot xmlns="http://endeca.com/schema/content/xtags/2010"
        zone="LeftColumnSlot" type="SidebarItem" maxContentItems="3"/>
```

```
    </Property>
  </ContentItem>
  <EditorPanel>
    <BasicContentItemEditor>
      <editors:DynamicSlotEditor propertyName="contents"
        zoneConfig="services/zones"/>
    </BasicContentItemEditor>
  </EditorPanel>
</ContentTemplate>
```

Creating templates for dynamic slot content

Standalone pieces of dynamic slot content in Experience Manager are based on page templates of a certain format.

Note: Although dynamic slot content is intended to be a section of a page rather than a full landing page of its own, the template must be of type `PageTemplate` in order for content administrators to assign a trigger to a piece of content in Experience Manager.

A page template intended for dynamic slots should contain exactly one content item; this content item is the dynamic slot content that is included in the page at runtime. A content administrator can specify multiple pieces of dynamic slot content to display on a page by creating several such pieces with the same or overlapping trigger criteria.

As a best practice, you should create one dynamic slot content template for each zone within your application, such as CenterColumnZone, LeftColumnZone, and RightColumnZone.

To create a page template for reusable dynamic slot content:

1. Create a template of type `PageTemplate`. For example:

   ```
   <ContentTemplate xmlns="http://endeca.com/schema/content-template/2008"
     type="PageTemplate" id="LeftColumnWidget">
     <RuleInfo zone="LeftColumnSlot" style="PageStyle"/>
     <Description>A widget to be placed in a left column dynamic content
   slot.</Description>
   </ContentTemplate>
   ```

   Note that the `<RuleInfo>` element is required because this is a page template. The `zone` attribute specifies the zone in which the dynamic slot content will be created.

   Because the zone is not visible in the Experience Manager interface, and because the zone for the dynamic slot content must match the zone specified in the Dynamic Slot cartridge in a page in order for the content to display, Oracle recommends that you give the page template an `id` that matches the name of the zone (in this case, an `id` of `LeftColumnWidget` for a template that has a `LeftColumnSlot` zone). This helps the content administrator identify the right template to select when creating the dynamic slot content.

2. Define a single content item in this template. For example:

   ```
     <ContentItem>
       <Name>Left Column Widget</Name>
       <Property name="widget">
         <ContentItem type="SidebarItem"/>
   ```

```
    </Property>
  </ContentItem>
```

The property name must be `widget` in order to enable the Dynamic Slot tag handler for the Content Assembler API.

The content item `type` must be the same as the type specified for this template's zone in the zone configuration file. For example, the `zones.xml` file for this application contains the following line:

```
  <Zone name="LeftColumnSlot" type="SidebarItem" maxItems="12"/>
```

Because the `zone` specified in the `<RuleInfo>` element of this template is `LeftColumnSlot`, the `type` of the content item named `widget` must be `SidebarItem`. It is important that the content item types match because Experience Manager cannot otherwise ensure that the content administrator is placing the correct type of cartridge in the slot.

3. There is no need to specify an editor in this template. Recall that the cartridge selector is enabled by default when you specify a content item property.

An example of a complete template is reproduced below:

```
<ContentTemplate xmlns="http://endeca.com/schema/content-template/2008"
  type="PageTemplate" id="LeftColumnWidget">
  <RuleInfo zone="LeftColumnZone" style="PageStyle"/>
  <Description>A widget to be placed in a left column dynamic content
slot.</Description>
  <ContentItem>
    <Name>Left Column Widget</Name>
    <Property name="widget">
      <ContentItem type="SidebarItem"/>
    </Property>
  </ContentItem>
  <EditorPanel>
    <BasicContentItemEditor/>
  </EditorPanel>
</ContentTemplate>
```

Creating renderers for dynamic slots

To enable the front-end application to handle dynamic slots, you must write a renderer for dynamic slot cartridges. Additionally, implementing the dynamic slots feature requires the use of a tag handler extension to the Content Assembler.

About the Dynamic Slot tag handler

Dynamic Slot tag handlers for the Content Assembler API are provided with Experience Manager Extensions. When the Content Assembler encounters a `<DynamicSlot>`, the tag handler retrieves the content resources in the specified zone for that slot. Recall that each piece of dynamic slot content represents a single content item (wrapped in a page template), and that several such content items can be returned for a given zone. From each of the content resources returned for the zone, the tag handler retrieves the content item in the property named `widget` and adds it to a content item list. The tag handler then returns this `ContentItemList` object as a property of the dynamic slot cartridge within the the overall landing page.

About the renderers

Your rendering code must be able to handle the content item list that is returned as part of the dynamic slot cartridge.

If you used the same content types for dynamic slot content as for regular cartridges in landing pages, this should be as simple as retrieving the appropriate property (in the case of the sample cartridge templates, this is named `contents`) and iterating through the list to dispatch the rendering to the specific code designed to render each specific content item type.

Sample renderers are provided with Experience Manager Extensions. As long as your dynamic slot cartridges share the same property name that contains the dynamic slot content, you do not need to write individual renderers for each dynamic slot cartridge.

Related Links

Deploying the Experience Manager Extensions reference application for Java on page 16
You can install the Experience Manager Extensions reference application into the Endeca Tools Service by using the sample context file included in the installation package.

Deploying the Experience Manager Extensions reference application for the RAD Toolkit for ASP.NET on page 16
You can install the Experience Manager Extensions reference application into an instance of IIS 6 or 7.

Working with dynamic slots in Experience Manager

Using the dynamic slots feature within Experience Manager is a two-part process: adding a placeholder for dynamic slot content on a page, and creating the pieces of dynamic slot content themselves.

Adding a dynamic slot cartridge to an existing page

A dynamic slot cartridge creates a placeholder for dynamic slot content on a particular landing page in Experience Manager. After you have set up placeholders on the pages in your site, you can manage the content that displays on those pages without editing the original page configuration.

To add a dynamic slot cartridge to a landing page:

1. Create or edit a landing page in a landing page zone.
2. Add a dynamic slot cartridge to one of the sections on the page. This is a cartridge that contains a Dynamic Slot editor.
3. In the Content Details Pane, the following parameters can be configured:

Option	Description
Rule Zone	The zone from which to fetch dynamic slot content for this cartridge. This must be a zone that was specified in the zone configuration file as containing the appropriate type of content item for this cartridge.
Max Content Items	The maximum number of dynamic slot items to display in this section for this page. The absolute limit is specified in the zone configuration file and the zone definition of the application's instance configuration, but this option enables you to limit it further on a per-page basis.

4. Save the changes to the page.

This completes the setup of the dynamic slot placeholder on a specific landing page. You then manage the content that populates the dynamic slot by creating standalone pieces of dynamic content. A dynamic slot placeholder can contain more than one piece of dynamic slot content, so you only need

to create one placeholder for each section of the page, rather than one for each cartridge that you want to manage as dynamic slot content.

Managing standalone dynamic slot content

Once the developer and content administrator setup is complete, you can manage the content that populates the dynamic slot cartridges on a page by creating and configuring standalone pieces of dynamic slot content.

Typically, dynamic slot content represents the aspects of the page that change frequently, while the overall landing page configuration remains relatively stable.

To create a standalone piece of dynamic slot content:

1. Navigate to the "landing page group" that corresponds to the zone for which you want to create dynamic slot content.

   The zone is configured in the dynamic slot cartridge that was added to a landing page.
2. Create a "New Landing Page."

   Although dynamic slot content is not really a landing page, Experience Manager only allows trigger criteria to be specified for landing pages, and not individual cartridges.
3. Select a "template" that is designed for dynamic slot content, rather than for whole landing pages.

   Important: Be sure to select the correct template. This should be a template that specifies the same zone as the dynamic slot cartridge that you want to populate. Experience Manager does not enforce which templates are available within a particular landing page group based on zone, so you must rely on naming conventions to select the right template.
4. Specify the trigger criteria for the dynamic slot content.
5. Insert a cartridge into the dynamic slot "page" and configure it as you normally would.
6. Save your dynamic slot content.
7. Optionally, adjust the priority of your new dynamic slot "page."

   Recall that more than one dynamic slot item can be displayed in the dynamic slot section. The priority determines the order in which the items are displayed.

Repeat these steps as many times as needed to cover all the desired permutations of dynamic slot content.

Chapter 5

Developing Editors with Experience Manager Extensions

This section provides instructions for setting up your development environment, including configuring the included Flex project.

Flex prerequisites and resources

You must set up a Flex development environment if you want to create your own community editors with Experience Manager Extensions.

Flex development requirements

In addition, the following software is required to build the source code projects:

- **Java SDK 1.5 or later.** — Java 1.6 is included with the Workbench 2.1 installation.
- **Apache Ant 1.6 or later.** — Apache Ant is an open source project hosted at *http://ant.apache.org*.
- **Flex SDK 3.2 or Flash Builder** — Flash Builder is an integrated editing and debugging environment offered by Adobe. It requires a developer license. The raw SDK is open source and also available from Adobe. You can consult the wiki at *http://opensource.adobe.com/wiki/display/flexsdk* for additional information.

Note: The instructions in this section assume that you are using the Flash Builder plugin for Eclipse.

Flex resources

Flex is an open source development framework created and maintained by Adobe. It supports common design patterns and is based on MXML and ActionScript 3.

If you are unfamiliar with Flex, you may find the following resources helpful:

- *http://www.adobe.com/products/flex/overview* — The Adobe Flex Web site provides an overview of the Flex development framework and includes download links to the Flex SDK.
- *http://www.adobe.com/support/documentation/en/flex* — The Adobe Flex resources page contains links to Flex documentation, including the *Adobe Flex Language Reference* and *Flex 3 Developer Guide*.

- *http://opensource.adobe.com/wiki/display/flexsdk/Coding+Conventions* — The Adobe Open Source wiki page on Flex SDK best practices and coding conventions provides a set of standards for coding in Flex.

Verifying the Flex SDK version in Flash Builder

If you choose to develop using Flash Builder, version 4.0 is required. In addition, the version of the Flex SDK that Flash Builder uses to compile must be exactly 3.2.

To verify the Flex SDK version:

1. Start Flash Builder.
2. Navigate to **Window** > **Preferences**.
3. In the tree control on the left, navigate to **Flex** > **Installed Flex SDKs**.
4. Verify that a "Flex 3.2" listing appears in the list of Installed SDKs.

Importing the Experience Manager Extensions project into Flash Builder

If you are using Flash Builder to build and debug Flex code, take the following steps to import the project into Flash Builder.

To import the Flex project:

1. Extract the contents of the Experience Manager Extensions source archive, `oexmx-<version>-source.zip`, into a new folder.

   This is your project folder.
2. Open Flash Builder.
3. Select **File** > **Import** > **Flex Project**.
4. Select **Import project from:** > **Project folder:**
5. Browse to your project folder.
6. Uncheck the **Use default location** checkbox.
7. Copy and paste the project folder path into the **Folder:** field.
8. Click **Finish**.

Creating the build.properties file

The included Ant scripts rely on a `build.properties` file to deploy the Experience Manager Extensions Web application to the correct URL. You must create this file in a `conf` subdirectory of your project folder.

This section assumes that you have extracted the Experience Manager Extensions source archive and created a folder for your Flash Builder project.

To create the `build.properties` file:

1. Create a subdirectory named `conf` in the project folder.
2. Create a new text file named `build.properties` in the `conf` folder.

This is a local property definition file for Ant builds.

3. Open the `build.properties` file.
4. To set the `FLEX_HOME` environment variable, add the following line:

   ```
   FLEX_HOME=<flex-install-path>
   ```

   Where *<flex-install-path>* is the absolute file path of your Flex SDK 3.2 installation. For example:

   ```
   FLEX_HOME=c:/flex_sdk_3.2
   ```

   Note: If you have installed Flash Builder, the SDK installation is nested within the Flash Builder installation.

5. To set the `deploy.url` property, add the following line:

   ```
   deploy.url=<deployment-path>
   ```

   Where *<deployment-path>* is the URL of your Web application. For example:

   ```
   deploy.url=http://<host>:<port>/oexmx.
   ```

6. Save and close the file.

Building and deploying the Experience Manager Extensions project

Experience Manager Extensions includes custom Ant tasks that handle most configuration and deployment automatically.

Once you have installed the Ant plugin for Flash Builder, you can execute Ant tasks without exiting your development environment.

Note: You can also build and deploy the project from the command line by navigating to the project folder in a command shell and running the `ant` command.

To build and deploy your project in Flash Builder:

1. Select the `pbx` project and choose the **Project** > **Build All** command.

   Alternately, you can use the **CTRL-B** keyboard shortcut.

2. (Optional) If you want to deploy without rebuilding, run the `"deploy"` target of the Ant script in the project folder.

   If you have installed the Ant plugin, you may do this from Flash Builder. Otherwise, run `ant deploy` from a command shell in the project folder. If you build the entire project from a command shell, you do not need to run the `deploy` task separately.

3. Verify your deployment by navigating to `http://<host>:<port>/oexmx/etc/pbconfig.xml`.

Setting up Experience Manager Extensions as an Eclipse project

You can set up an Eclipse environment for developing editors with Experience Manager Extensions by following the steps below.

Setting up Experience Manager Extensions as an Eclipse project requires the Flash Builder 4 plugin for Eclipse.

To set up Experience Manager Extensions as an Eclipse project:

1. Navigate to your Experience Manager Extensions installation directory.
2. Extract the ZIP archive.
3. Select **File** > **New** > **Project** > **Flex Project**.
4. In the **Project name:** field, enter `pbx`.
5. Under **Contents**, select the **Create project from existing source** radio button.
6. Enter or browse to the directory where you extracted your source archive in Step 2.
7. In the tree control, browse to `launcher\src\main\flex`.
8. Right-click and select **Use as Source Folder**.
9. In the tree control, browse to `launcher\src\main\flex\launcher.mxml` and set it as the Main Application File.
10. Select the top-level `pbx` directory in the tree control and click **Link Additional Source to Project...**
11. In the **Linked Folder Location** field, enter or browse to `aslib\src\main\flex`.
12. Click **Next**.
13. Click **Finish**.
14. Repeat Steps 10-13 for the following directories:
    - `editors\src\main\flex`
    - `xtags\src\main\java`
15. Click **Finish**.

If you have not deployed Experience Manager Extensions alongside your Workbench installation or as a standalone Web application, do so now.

Once you have deployed Experience Manager Extensions, you can build from source and deploy the changes to the running Experience Manager Extensions instance by right-clicking `build.xml` in Eclipse and selecting **Run As** > **Ant Build**.

Related Links

Flex prerequisites and resources on page 61
You must set up a Flex development environment if you want to create your own community editors with Experience Manager Extensions.

Importing the Experience Manager Extensions project into Flash Builder on page 62
If you are using Flash Builder to build and debug Flex code, take the following steps to import the project into Flash Builder.

Basic sample editors on page 65
Several open source editors are provided to supplement or replace Experience Manager's standard editors. These basic editors can be easily combined to create more complex editing interfaces.

Testing and debugging cartridge modifications on page 66
You can edit the included template files or create your own to test your new cartridges and sample cartridge modifications.

Basic sample editors

Several open source editors are provided to supplement or replace Experience Manager's standard editors. These basic editors can be easily combined to create more complex editing interfaces.

Editor	Use
`BooleanEditor`	A check box for editing a true/false value.
`CalendarEditor`	An editor that allows you to select a date and time value, returning a String.
`ColorEditor`	A visual color selector for editing an RGB string value.
`ChoiceEditor`	A drop-down menu offering a choice of string values.
`DataManipula¬tionEditor`	An editor that allows you to enter in a series of strings, which are returned as a comma-separated list.
`RefinementSelec¬torEditor`	By configuring a dimension ID in the cartridge, this editor allows you to select a dimension value.
`RichTextEditor`	An update of the `RichTextEditor` provided with the Experience Manager samples. This version is greatly simplified through the use of common components and utilities.
`StringEditor`	A single- or multiline text input box for editing a string value.

Basic editor configuration

Most of the Experience Manager Extensions basic editors have a common set of configuration options.

You can refer to the Open Editor Test template for examples of how to configure the sample editors. The template is located in the Experience Manager Extensions source zip under `content\templates\MainColumnContent-OpenEditorsTest.xml`.

Common configuration

The basic editors share the following attributes:

Attribute	Description
`propertyName`	Required. The `name` of the property that this editor is associated with. This property must be declared in the same template as the editor.
`enabled`	If set to `false`, the editor displays in Experience Manager but the value cannot be changed by the user. By default, editors are enabled.

Attribute	Description
`label`	This attribute allows you to specify a more descriptive label for this field in Experience Manager. If no label is specified, the property name is used by default.

Choice editor configuration

The `<ChoiceEditor>` takes the following additional attributes:

Attribute	Description
`editable`	If set to true, this attribute allows Experience Manager users to specify custom string values. Default value is false.
`prompt`	Specifies a custom prompt, such as "Select an option." Default value is an empty string.
`tooltip`	If present, specifies optional help text to display in a tool tip window. Default is no tool tip.

String editor configuration

The `<StringEditor>` takes the following additional attributes:

Attribute	Description
`rowProperty`	A number representing the number of rows. If the value of this property is greater than `1`, the editor is a text area instead of a single-row text field. A text area has word wrap enabled, whereas a single-row field does not.
`width`	The width in pixels of the text field presented in the Experience Manager interface. The default width is 200 pixels.
`height`	The height in pixels of the text field presented in the the Experience Manager interface. The default height for a single-row field is 24 pixels. Set this to a higher value for a multiline text area.

Testing and debugging cartridge modifications

You can edit the included template files or create your own to test your new cartridges and sample cartridge modifications.

In addition to modifying template files, you must also add any new editors to the `pbconfig.xml` file.

About modifying the editor configuration file

To add new editors to your installation of Experience Manager, you must modify the editor configuration file, `pbconfig.xml`.

The `pbconfig.xml` file is generated from `xml.jsp`. In order to generate the XML file from the edited JSP, you must alter the file in your Flash Builder project directory and rebuild the Experience Manager Extensions project.

Because the JSP creates a configuration file at build time, some of the syntax is altered from the XML described in the *Oracle Endeca Experience Manager Developer's Guide*. The `url` attribute of the `<EditorModule>` element points to the `editors.swf` output location from the project build, so any editors that you wish to add to the project must be part of the same module.

To add additional editors, insert an `<Editor>` element inside the `<EditorModule>` element for each new editor that you wish to include. Use the following syntax:

```
...
<EditorConfig xmlns="http://endeca.com/schema/editor-config/2010" >
    <GlobalEditorConfig>
        <pbxUrl><%= getPbxUrl(request) %></pbxUrl>
    </GlobalEditorConfig>

    <EditorModule url="<%= getPbxUrl(request) %>/modules/editors.swf">
        <Editor name="editors:MyCustomEditor">
            <EditorConfig/>
        </Editor>
    </EditorModule>
</EditorConfig>
```

For detailed information on modifying editor configuration information and adding editor entries to the editor configuration file, please refer to the *Oracle Endeca Experience Manager Developer's Guide*.

Index

S

T

V

Z