Oracle Endeca Commerce

Experience Manager Editor Developer's Guide

Version 3.1.0 • July 2012

Contents

Preface.....7
About this guide.....7
Who should use this guide.....7
Conventions used in this guide.....8
Contacting Oracle Support.....8

Chapter 1: Setting up the Experience Manager SDK.....9
Flex prerequisites and resources.....9
About setting up a Flex development environment.....9
    Verifying the Flex SDK version in Flash Builder.....10
Installing the Experience Manager API.....10

Chapter 2: Developing Editors With the Experience Manager Editor SDK.11
About developing custom editors.....11
The sample RichTextEditor.....11
    The sample Rich Text Box cartridge template.....14
    Installing the sample editor module.....14
Including custom editors in the sample editor module.....16
Registering custom editors.....17
About creating and uploading a cartridge template.....18

Copyright and disclaimer

Preface

Oracle Endeca's Web 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 developing extensions for Experience Manager using the Experience Manager Extension SDK.

This guide assumes that you have read the *Assembler Application Developer's Guide* and that you are familiar with Experience Manager in Workbench.

Who should use this guide

This guide is intended for developers who are familiar with Oracle Endeca Commerce Tools and Frameworks and who want to use the Experience Manager Editor SDK as an additional development resource.

If you are not familiar with using Experience Manager, Oracle recommends reading the *Assembler Application Developer's Guide* prior to this guide.

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

Setting up the Experience Manager SDK

This section provides instructions for setting up your development environment for developing custom editors.

Flex prerequisites and resources

The Experience Manager Editor SDK requires the following components:

Flex development requirements

- **Apache Maven 3.0.4** — Apache Maven is an open source project hosted at *http://maven.apache.org/*.
- **Flex SDK 4.5 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://sourceforge.net/adobe/flexsdk/wiki/About/* for additional information.

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.html* — The Adobe Flex Web site provides an overview of the Flex development framework and includes download links to the Flex SDK.
- *http://help.adobe.com/en_US/Flex/4.0/UsingSDK/index.html* — The Adobe Flex 4 resources page contains links to Flex documentation, including the *ActionScript 3.0 Reference* and *ActionScript 3.0 Developer's Guide*.

About setting up a Flex development environment

Oracle recommends setting up a Flex development environment for creating your custom editors.

You can use a standalone installation of Flash Builder, or set up your development environment in Eclipse. Setting up an Experience Manager SDK project in Eclipse requires the Flash Builder 4 plugin.

Verifying the Flex SDK version in Flash Builder

If you choose to develop using Flash Builder, you must use Flash Builder 4.0. In addition, the version of the Flex SDK that Flash Builder uses to compile must be exactly 4.5.

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 4.5" listing appears in the list of **Installed SDKs**.

Installing the Experience Manager API

The Experience Manager Editor SDK includes a Maven script for installing the included API package into your Maven repository.

To install the Experience Manager API package:

1. Navigate to the `%ENDECA_TOOLS_ROOT%\editor_sdk\reference\build\maven` directory.
2. Open a command prompt.
3. Run the `install-api` profile in the Maven POM file using the following command:

   ```
   mvn -Pinstall-api
   ```

   This installs `ExperienceManagerAPI-3.1.0.swc` to your Maven repository.

Chapter 2

Developing Editors With the Experience Manager Editor SDK

This chapter covers steps for developing your own editors using the Experience Manager Editor SDK.

About developing custom editors

Once you have set up your development environment, creating and using new editors consists of the following general steps:

1. Build your editor module and copy it to your `<App_Dir>\config\editors_config\modules` directory.
2. Register your custom editor module and the included editors within your application's editor configuration file.
3. Upload the editor module and editor configuration file to your deployed application by using the `<App_Dir>\control\set_editors_config` script.
4. Add your editors to an existing cartridge template or create a new cartridge template that includes them.

   

   Note: If you make changes to an existing cartridge, any saved cartridges that use the old template are unchanged until you access them in Experience Manager. When accessed, they are updated with default values specified in the cartridge template. For this reason, Oracle does not recommend updating cartridges that are in widespread use throughout your application.

5. Upload the cartridge template to your deployed application by using the `<App_Dir>\control\set_templates` script.

The sample RichTextEditor

The Experience Manager editor SDK includes a sample `RichTextEditor` and associated cartridge template that you can use as a reference when developing your own editors.

The source code for the `RichTextEditor` is available under *editor_sdk\reference\build\maven\src\main\flex\com\endeca\tools\pagebuilder\samples\editors\RichTextEditor.mxml* and is reproduced below:

```
<ext:Editor direction="horizontal"
    xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:ext="com.endeca.tools.pagebuilder.editor.*"
    editorReady="editorReadyHandler(event)">
  <mx:Script>
    <![CDATA[
      import mx.utils.StringUtil;
      import com.endeca.tools.pagebuilder.events.EditorEvent;
      import com.endeca.tools.pagebuilder.model.content.IProperty;
      import com.endeca.tools.pagebuilder.model.content.IContentItem;
      import com.endeca.tools.pagebuilder.model.content.IStringProperty;

      private static const DEFAULT_WIDTH:int  = 300;
      private static const DEFAULT_HEIGHT:int = 400;

      private var _property:IStringProperty;

      [Bindable]

      /**
       * The property which contains the string that we are using as the
       * data source for this editor
       */
      protected function get property():IStringProperty
      {
        return _property;
      }

      protected function set property(value:IStringProperty):void
      {
        _property = value;
      }

      /**
       * The "editorReady" event is dispatched on an editor after all
       * properties from the <code>IEditor</code> interface have been set
on
       * it.  You can listen for this event if you want to do things at
this
       * stage in the editor lifecycle.  Typical operations might include

       * getting/validating specific properties from the content item.
       */
      private function editorReadyHandler(event:EditorEvent):void
      {
        // Get the property you are using to store Rich Text information
        if (contentItem != null && info.templateConfig != null)
        {
          /*
           * The property name is stored in the propertyName attribute of
the templateConfig (in
           * this case, the RichTextEditor element in the page template).
In the
           * sample template the RichTextEditor templateConfig looks like
this:
           *
           * <RichTextEditor propertyName="rich_text" xmlns="http://ende¬
```

```
ca.com/sample/2010" />
           */
          var propertyName:String = info.templateConfig.@propertyName;
         var prop:IProperty = contentItem.getPropertyByName(propertyName);


          /*
           * Make sure we are working with the right type of property. The
 rich_text Property
           * in the sample page template is an IStringProperty:
           *
           * <Property name="rich_text">
           *   <String />
           * </Property>
           */
          if(prop == null)
          {
            error('Property not found: "{0}"', [propertyName]);
          }
          // this editor works with String properties only
          else if(!(prop is IStringProperty))
          {
            var params:Array =
              [flash.utils.getQualifiedClassName(prop),
               flash.utils.getQualifiedClassName(IStringProperty)];

            error('Property type mismatch: "{0}"; expected "{1}', params);

          }
          else
          {
            property = IStringProperty(prop);

            propertyValueTextInput.enabled =
             info.templateConfig.@enabled != 'false' && !property.readOnly;


            var width:int  = new int(info.templateConfig.@width);
            var height:int = new int(info.templateConfig.@height);

          propertyValueTextInput.width = width > 0 ? width : DEFAULT_WIDTH;

            propertyValueTextInput.height = height > 0 ? height : DE¬
FAULT_HEIGHT;
          }
        }
      }

      /**
       * Constructs an error message from the given message and parameters

       * and then dispatches an "editorError" event.
       */
     private function error(message:String, messageParams:Array=null):void

      {
        var msg:String = StringUtil.substitute(message, messageParams);
        dispatchEvent(EditorEvent.createErrorEvent(msg));
      }
    ]]>
  </mx:Script>
```

```
  <mx:FormItem id="propertyItem"
      label="{info.templateConfig.@label}"
      labelWidth="135"
      labelStyleName="leftColumnLabel"
      required="false">
    <mx:RichTextEditor id="propertyValueTextInput"
        htmlText="{property.string}"
        change="property.string = propertyValueTextInput.htmlText;"
        borderThickness="3"
        fontFamily="Verdana"/>
  </mx:FormItem>
</ext:Editor>
```

The sample Rich Text Box cartridge template

The Rich Text Box cartridge is provided as a sample cartridge that makes use of the `RichTextEditor`. The associated cartridge template is included with the Experience Manager Editor SDK.

The cartridge template is available under `editor_sdk\reference\cartridge_templates\SecondaryContent-SampleTemplate.xml`. The cartridge uses a basic `StringEditor` for the title box, and the `RichTextEditor` to enter and configure body text:

```
<ContentTemplate xmlns="http://endeca.com/schema/content-template/2008"
type="SecondaryContent" id="SampleEditor">
  <Description>Displays rich text.</Description>
   <ThumbnailUrl>/thumbnails/PageTemplate/TextBoxSidebar.png</ThumbnailUrl>

  <ContentItem>
    <Name>New Rich Text Box</Name>
    <Property name="title">
      <String/>
    </Property>
    <Property name="rich_text">
      <String/>
    </Property>
  </ContentItem>

  <EditorPanel>
    <BasicContentItemEditor>
      <StringEditor propertyName="title" label="Title" enabled="true"
xmlns="editors"/>
     <!-- default value for the optional height attribute for RichTextEditor
 is 400px -->
     <!-- make sure not to make it any smaller or it will not render well
 -->
     <RichTextEditor propertyName="rich_text" xmlns="http://endeca.com/sam¬
ple/2010" label="Custom Editor" />
    </BasicContentItemEditor>
  </EditorPanel>
</ContentTemplate>
```

Installing the sample editor module

The Experience Manager Editor SDK includes a sample editor module with a Rich Text editor that you can install in your application.

Note: For example purposes this guide assumes that you are extending the default Discover Electronics reference application, with Workbench running on port `8006` of your local machine.

To install the sample editor module:

1. Create a directory for custom editor modules:
   a) Navigate to the `<App_Dir>\config\editors_config` directory.

      For the default Discover Electronics reference application, this is `C:\Endeca\apps\Discover\config\editors_config` on Windows, or `/usr/local/endeca/apps/Discover/config/editors_config` on UNIX.
   b) Create a `modules` directory.
2. Build the sample editor module:
   a) Navigate to the `%ENDECA_TOOLS_ROOT%\editor_sdk\reference\build\maven` directory.
   b) Build the sample editor module using the following command:

      ```
      mvn clean install
      ```

      The `sample_editors.swf` file is output to the `target` subdirectory.
   c) Navigate to the `%ENDECA_TOOLS_ROOT%\editor_sdk\reference\build\maven\target` directory.
   d) Copy the `sample_editors.swf` file to the `<App_Dir>\config\editors_config\modules` directory you created in Step 1.
3. Register the sample editors:
   a) Open the sample editor configuration file, `editor_sdk\reference\editors_config\sample_editors.xml`.
   b) Open the editor configuration file for your application.

      For the default Discover Electronics reference application, this is `C:\Endeca\apps\Discover\config\editors_config\editors.xml` on Windows, or `/usr/local/endeca/apps/Discover/config/editors_config/editors.xml` on UNIX. .
   c) Copy the `<EditorModule>` element from the `sample_editors.xml` file to the editor registry file:

      ```
      <EditorModule url="/ifcr/sites/[site name]/configuration/tools/xmgr/mod¬
      ules/sample_editors.swf">
          <Editor name="http://endeca.com/sample/2010:RichTextEditor">
              <EditorConfig sample="customAttribute">
                  <ExampleCustomEditorXML foo="bar" size="10" resize¬
      able="false" />
              </EditorConfig>
          </Editor>
      </EditorModule>
      ```

      The element should be parallel to the existing `<EditorModule>` element.
   d) Replace `[site name]` with the name of your application:

      ```
      <EditorModule url="/ifcr/sites/[site name]/configuration/tools/xmgr/mod¬
      ules/sample_editors.swf">
      ```

      For the default Discover Electronics reference application, this is `Discover`.
   e) Save and close the file.
4. Upload your custom content:
   a) Navigate to the `<App_Dir>\control` directory.

b) Run the `set_editors_config` batch or shell script.
This uploads the `sample_editors.swf` and `editors.xml` files to the Endeca Configuration Repository.

5. Upload a template that includes the custom editor:
   a) Copy `editor_sdk\reference\cartridge_templates\SecondaryContent-SampleTemplate.xml` to your `<App_Dir>\config\cartridge_templates` directory.
   b) Navigate to the `<App_Dir>\control` directory.
   c) Run the `set_templates` batch or shell script.
   This uploads the template files to the Endeca Configuration Repository.

Including custom editors in the sample editor module

In order to add your editors to the `sample_editors.swf` file, you must modify the `sample_editors.mxml` editor registry to include them.

In order to build your custom editor module using the included Maven build, you must navigate to the `editor_sdk\reference\build\maven` directory and enter the following in your command prompt or shell:

```
mvn clean install
```

The resulting `sample_editors.swf` file is output to the `%ENDECA_TOOLS_ROOT%\editor_sdk\reference\build\maven\targetdirectory`. In order to upload it to your deployed application, you must copy the editor module to the `<App_Dir>\config\editors_config\modules` directory and run the `<App_Dir>\control\set_editors_config` batch or shell script.

To include custom editors in the sample editor module:

1. Navigate to the `editor_sdk\reference\build\maven\src\main\flex` directory.
2. Open `sample_editors.mxml`.
3. Within the `<mx:Array>` element, add an `<editor:EditorEntry >` element for each of your custom editors:

```
<editor:EditorModule xmlns:mx="http://www.adobe.com/2006/mxml"
  xmlns:editor="com.endeca.tools.pagebuilder.editor.*">
  <editor:registeredEditors>
    <mx:Array>
      <editor:EditorEntry
        uri="http://endeca.com/sample/2010"
        localName="RichTextEditor"
       editor="com.endeca.tools.pagebuilder.samples.editors.RichTextEd¬
itor" />
      <editor:EditorEntry/>
    </mx:Array>
  </editor:registeredEditors>
</editor:EditorModule>
```

4. Specify the following attributes:

Attribute	Value
`uri`	The editor namespace. This is used in your cartridge templates and in the editor configuration file.

Attribute	Value
localName	The name to use for the editor in your cartridge templates.
editor	The fully qualified name of your editor.

```
<editor:EditorModule xmlns:mx="http://www.adobe.com/2006/mxml"
  xmlns:editor="com.endeca.tools.pagebuilder.editor.*">
  <editor:registeredEditors>
    <mx:Array>
      <editor:EditorEntry
        uri="http://endeca.com/sample/2010"
        localName="RichTextEditor"
       editor="com.endeca.tools.pagebuilder.samples.editors.RichTextEd¬
itor" />
      <editor:EditorEntry
        uri="http://endeca.com/sample/2010"
        localName="MyCustomEditor"
       editor="com.endeca.tools.pagebuilder.samples.editors.MyCustomEd¬
itor" />
    </mx:Array>
  </editor:registeredEditors>
</editor:EditorModule>
```

5. Save and close the file.
6. Build the sample editor module:
   a) Navigate to the %ENDECA_TOOLS_ROOT%\editor_sdk\reference\build\maven directory.
   b) Build the sample editor module using the following command:
      ```
      mvn clean install
      ```
      The sample_editors.swf file is output to the target subdirectory.
   c) Navigate to the %ENDECA_TOOLS_ROOT%\editor_sdk\reference\build\maven\target directory.
   d) Copy the sample_editors.swf file to the <App_Dir>\config\editors_config\modules directory.
      If this directory does not exist, you must create it.

Registering custom editors

You must modify the editor configuration file to register new editors with Experience Manager. The file, editors.xml, is maintained in the Endeca Configuration Repository and stored locally within the <App_Dir>\config\editors_config directory.

Note: The steps below assume a default installation, with Workbench running on port 8006 of your local machine.

To register a custom editor:

1. Navigate to the <App_Dir>\config\editors_config directory of your deployed application.
   For the default Discover Electronics reference application, this is C:\Endeca\apps\Discover\config\editors_config on Windows, or /usr/local/endeca/apps/Discover/config/editors_config on UNIX.

2. Open the `editors.xml` file.
3. Add an `<EditorModule>` element within the closing `</EditorConfig>` tag, and set the `url` attribute to your custom editor module.

   For example:

```
    ...
   <EditorModule url="/ifcr/sites/Discover/configuration/tools/xmgr/mod¬
ules/sample_editors.swf">
    </EditorModule>
</EditorConfig>
```

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

   For example:

```
    ...
   <EditorModule url="/ifcr/sites/Discover/configuration/tools/xmgr/mod¬
ules/sample_editors.swf">
        <Editor name="http://endeca.com/sample/2010:RichTextEditor">
        </Editor>
    </EditorModule>
</EditorConfig>
```

5. To set default editor configuration, add an `<EditorConfig>` element within each `<Editor>` element:

   For example:

```
    ...
   <EditorModule url="/ifcr/sites/Discover/configuration/tools/xmgr/mod¬
ules/sample_editors.swf">
        <Editor name="http://endeca.com/sample/2010:RichTextEditor">
            <EditorConfig sample="customAttribute">
                <ExampleCustomEditorXML foo="bar" size="10" resize¬
able="false" />
            </EditorConfig>
        </Editor>
    </EditorModule>
</EditorConfig>
```

6. Save and close the file.
7. Navigate to the `<App_Dir>\control` directory of your deployed application.
8. Run the `set_editors_config` batch or shell script.
   This script uploads the updated `editors.xml` file and any editor modules in the `<App_Dir>\config\editors_config\modules` directory to the Endeca Configuration Repository.

About creating and uploading a cartridge template

To use your custom editors in Experience Manager, you need to create and upload a cartridge template that includes the new editors. You can choose to create a new cartridge, or to modify an existing cartridge template.

After creating or modifying a cartridge to include your custom editors, you must upload it to your application.. You can accomplish this by moving the template to your deployed application's

`config\cartridge_templates` directory and running the `control\set_templates` batch or shell script.

Note: The *Assembler Application Developer's Guide* includes detailed information on creating cartridge templates.

Index

A

Adobe Flex
    verifying version 10
Apache Maven 10

C

cartridge template 18

D

development environment 9

E

editor configuration file 17
editor modules
    building 16
editors
    building 16
    developing 11, 16
    registering 17
Experience Manager API
    installing 10

P

prerequisites
    Adobe Flex 9
    Apache Maven 9

S

sample editor
    building 15
    registering 15
    RichTextEditor 12
    sample cartridge 14
sample editor module 15

W

workflow 11