Oracle Endeca Commerce

Tools and Frameworks Migration Guide

Version 3.1.1 · December 2012

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 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 how to upgrade earlier versions of Oracle Endeca Commerce Tools and Frameworks to the most recent version, and the guide also describes how to migrate an Endeca application to the most recent version of Tools and Frameworks.

Who should use this guide

This guide is intended for application developers who are using Oracle Endeca Commerce Tools and Frameworks and are responsible for migration tasks.

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

Introduction

This section contains basic information about the migration process.

Recommended reading

In addition to reading this document, Oracle recommends that you read the following documents for important information about the release.

Release Announcement

The Release Announcement provides an explanation of the new features that were added in the latest version. The Release Announcement is available as part of the Tools and Frameworks documentation set that you can download from the Oracle Technology Network.

Release Notes

The Release Notes (README.txt) provide information about bug fixes and known issues for this release. The Release Notes are installed into `ToolsAndFrameworks/<version>`.

Note: Although the release notes are available with the installation packages, the latest versions (and possible revisions) of release notes for each package are also available on the Oracle Technology Network.

Oracle Endeca Commerce Getting Started Guide

The *Oracle Endeca Commerce Getting Started Guide* gives an overview of Endeca components and provides setup and operations instructions for a single machine environment. You can download the *Oracle Endeca Commerce Getting Started Guide* from the Oracle Technology Network.

Oracle Endeca Commerce Migration Guides

In addition to the *Oracle Endeca Tools and Frameworks Migration Guide*, you may also need to upgrade other components of Oracle Endeca Commerce. The migration paths for each component are documented in the following guides:

- *Oracle Endeca MDEX Engine Migration Guide*
- *Oracle Endeca Platform Services Migration Guide*
- *Oracle Endeca Content Acquisition System Migration Guide*

Each guide is available on the Oracle Technology Network

Package compatibilities

To determine the compatibility of Tools and Frameworks with other Endeca software packages, see the *Oracle Endeca Commerce Compatibility Matrix* available on the Oracle Technology Network.

Prerequisite upgrades

Before upgrading to Tools and Frameworks 3.1.1, you must first upgrade the MDEX Engine 6.2.x to MDEX Engine 6.4.0. For details about this task, see the *MDEX Engine Migration Guide*.

Identifying your upgrade path

There are three supported upgrade paths:

- From Oracle Endeca Tools and Frameworks 3.1.0.1 to Oracle Endeca Tools and Frameworks 3.1.1
- From Oracle Endeca Tools and Frameworks 3.1.0 to Oracle Endeca Tools and Frameworks 3.1.1
- From Endeca IAP 2.1.2 (and its supporting components) to Oracle Endeca Tools and Frameworks 3.1.1

Upgrading from Tools and Frameworks 3.1.0.1 to 3.1.1

The 3.1.0.1 to 3.1.1 upgrade path is the most direct, as it encompasses only the differences between the 3.1.1 release and the release immediately preceding it.

Upgrading from Tools and Frameworks 3.1.0 to 3.1.1

In the 3.1.0 to 3.1.1 path, the upgrade path is straightforward because there are no significant architectural changes between versions.

Upgrading from Endeca IAP 2.1.2 to Tools and Frameworks 3.1.1

The following table lists the specific 2.1.2 component combinations you may have and provides information about upgrading to the latest version of Tools and Frameworks.

2.1.2 Components:	3.1.1 Components:
• Oracle Endeca Guided Search 2.1.2 (Workbench with Rule Manager) • Oracle Endeca Experience Manager 2.1.2 (Workbench with Page Builder) • Oracle Endeca Content Assembler API (Java or .NET) 2.1.2 • URL Optimization API 2.1 (Java or .NET) • Oracle Endeca Deployment Template 3.2.2	Oracle Endeca Tools and Frameworks 3.1.1

Unsupported upgrade paths

There are several unsupported upgrade paths:

- Endeca IAP 2.1.2 (and its supporting components) to Oracle Endeca Tools and Frameworks 3.0.0
- Endeca IAP 2.1.2 (and its supporting components) to Oracle Endeca Tools and Frameworks 3.1.0
- Oracle Endeca Tools and Frameworks 3.0.0 to Oracle Endeca Tools and Frameworks 3.1.0

Upgrading on a single host

For the sake of simplicity, this guide describes how upgrade the software and migrate an Endeca application on a single machine. This makes installation, configuration, and communication among components simpler as you migrate application configuration.

Chapter 2

Migrating an Endeca Application from 3.1.0.1 to 3.1.1

About upgrading your Tools and Frameworks installation

To upgrade to Tools and Frameworks 3.1.1, back up your existing Workbench and application configuration, delete your current installation of Tools and Frameworks, and install version 3.1.1.

Backing up the Workbench configuration files

Oracle Endeca Workbench uses several configuration files located in `%ENDECA_TOOLS_CONF%\conf` (on Windows) or `$ENDECA_TOOLS_CONF/conf` (on UNIX) to customize the behavior of various aspects of Workbench.

These files store Workbench configuration, user authentication configuration, and definitions of the menus and extensions in Workbench. If you have manually modified any of the following files from their default state, you should copy them to a backup location:

File name	Description
`Login.conf`	Configuration for user authentication using LDAP
`webstudio.properties`	Miscellaneous configuration parameters for Workbench
`webstudio.log4j.properties`	Configuration for the Workbench system log and audit log
`ws-extensions.xml`	Definitions of Workbench extensions
`ws-mainMenu.xml`	Definitions of the Workbench navigation menu and launch page

File name	Description
`ws-roles.xml`	This file is deprecated in the 3.1.1 release. User configuration for Workbench is managed in the Endeca Configuration Repository. For details, see the *Workbench Administrator's Guide*.

Backing up the application state

The `webstudiostore` and `emanager` directories contain information about your application state, including user and permission settings, preview application settings, content XML, and resource metadata.

To back up the application state directories:

1. Stop the Endeca Tools Service.
2. Copy the `webstudiostore` directory and its subdirectories from `%ENDECA_TOOLS_CONF%\state\` (on Windows) or `$ENDECA_TOOLS_CONF/state/` (on UNIX) to another location.

   This directory contains information such as users and permissions, as well as preview application settings.
3. Copy the `emanager` directory and its subdirectories from `%ENDECA_TOOLS_CONF%\state\` (on Windows) or `$ENDECA_TOOLS_CONF/state/` (on UNIX) to another location.

   This directory contains resource metadata, state information, and content XML.
4. Start the Endeca Tools Service.

Backing up application content from the Endeca Configuration Repository

You back up application configuration in the Endeca Configuration Repository using the `export_site` script provided with the Deployment Template.

The script exports application configuration in a format that can be re-imported to the Endeca Configuration Repository. The script connects to the Endeca Configuration Repository instance for the current application based on the configuration in `AppConfig.xml`.

Important: To guarantee consistent data, ensure that no baseline or partial updates are running during the backup process.

To back up application configuration in the Endeca Configuration Repository:

1. Navigate to the `control` directory of your deployed application, for example, `C:\Endeca\apps\discover\control`.
2. Run the `export_site` script, passing in an optional name for the export file, as in the following examples:

   On Windows:

   ```
   export_site.bat ..\ECR-backups\20121221.xml
   ```

On UNIX:

```
./export_site.sh ../ECR-backups/20111221.xml
```

If no file name is provided, it defaults to a file named according to the pattern `<site-name>-DD-MM-YYYY.xml` in the working directory.

Upgrading from Tools and Frameworks 3.1.0.1 to 3.1.1

Before installing Tools and Frameworks 3.1.1, you must back up your existing Workbench and application configuration.

To upgrade from Tools and Frameworks 3.1.0.1 to 3.1.1:

1. Uninstall Tools and Frameworks 3.1.0.1. See the *Oracle Endeca Commerce Tools and Frameworks Installation Guide* and perform the tasks in "Uninstalling Oracle Endeca Tools and Frameworks".
2. Install Tools and Frameworks 3.1.1. See the *Oracle Endeca Commerce Tools and Frameworks Installation Guide* and perform the tasks in "Installing Oracle Endeca Tools and Frameworks".

Once you have upgraded your installation of Tools and Frameworks, you can migrate the applications you previously backed up.

About migrating an Endeca application from 3.1.0.1 to 3.1.1

To migrate an Endeca application from 3.1.0.1 to 3.1.1, you must restore the application configuration from backups and publish the changes to the MDEX Engine and Endeca Configuration Repository.

Restoring a backup of the Workbench configuration files

You restore your Workbench configuration directory, `%ENDECA_TOOLS_CONF%\conf`, by merging in changes from your backup files into the new installation.

To restore a backup of the Workbench configuration files:

1. Stop the Endeca Tools Service.
2. Open your configuration backup files.
3. Manually merge any configuration changes into the files located at `%ENDECA_TOOLS_CONF%\conf` (on Windows) or `$ENDECA_TOOLS_CONF/conf` (on UNIX).

   Important: The syntax for the `Login.conf` file and tool IDs in `ws-mainMenu.xml` have changed for this release. See the *Workbench Administrator's Guide* for the current syntax.

4. Save and close the files.
5. Start the Endeca Tools Service.

Restoring a backup of the application state

You restore backups of the application state directories, `webstudiostore` and `emanager`, by copying backups into the new installation.

To restore backups of the application state directories:

1. Stop the Endeca Tools Service.
2. Delete the `webstudiostore` directory from `%ENDECA_TOOLS_CONF%\state\` (on Windows) or `$ENDECA_TOOLS_CONF/state/` (on UNIX).
3. Copy the backup of the `webstudiostore` directory, including all its subdirectories, to `%ENDECA_TOOLS_CONF%\state` (on Windows) or `$ENDECA_TOOLS_CONF/state/` (on UNIX).
4. Delete the `emanager` directory from `%ENDECA_TOOLS_CONF%\state\` (on Windows) or `$ENDECA_TOOLS_CONF/state/` (on UNIX).
5. Copy the backup of the `emanager` directory and its subdirectories to `%ENDECA_TOOLS_CONF%\state\` (on Windows) or `$ENDECA_TOOLS_CONF/state/` (on UNIX).
6. Start the Endeca Tools Service.

Restoring a backup of application configuration to the Endeca Configuration Repository

You restore a backup of your application configuration to an instance of the Endeca Configuration Repository by using the `import_site` script provided with the Deployment Template.

The script takes a file created by a previous export and imports its content to the Endeca Configuration Repository. The script connects to the Endeca Configuration Repository instance for the current application based on the configuration in `AppConfig.xml`.

Important: To guarantee consistent data, ensure that no baseline or partial updates are running during the backup process.

To restore a backup of your application configuration in the Endeca Configuration Repository:

1. Navigate to the `control` directory of your deployed application, for example, `C:\Endeca\apps\Discover\control`.
2. Run the `import_site` script, passing in the file name of the backup, as in the following examples:

   On Windows:

   ```
   import_site.bat discover-21-12-2012.xml
   ```

   On UNIX:

   ```
   ./import_site.sh discover-21-12-2012.xml
   ```

3. Run the `load_baseline_test_data` script.
4. Run the `baseline_update` script to publish updated information to the MDEX Engine.
5. Run the `set_templates` script to push updated templates to the repository.
6. Run the `promote_content` script.

Updating Endeca application JAR files to version 3.1.1

After restoring your content, you must update your application with 3.1.1 JAR files.

To update Endeca application JAR files:

1. Navigate to the `%ENDECA_TOOLS_ROOT%\3.1.1\deployment_template\app-templates\common\config\lib\java`

folder (on Windows) or `$ENDECA_TOOLS_ROOT/3.1.1/deployment_template/app-templates/common/config/lib/java` (on UNIX).

2. Copy the following files:
   - `commons-io-1.4.jar`
   - `commons-lang-2.4.jar`
   - `eacComponents-3.1.1.jar`
   - `eacToolkit-3.1.1.jar`
3. Navigate to the `config\lib\java` folder of your application, for example:
   `C:\Endeca\apps\Discover\config\lib\java`
4. Paste the JAR files that you copied into the folder.
5. Delete the previous versions of the JAR files:
   - `eacComponents-3.1.0.jar`
   - `eacToolkit-3.1.0.jar`

Chapter 3

Migrating an Endeca Application from 3.1.0 to 3.1.1

About upgrading your Tools and Frameworks installation

To upgrade to Tools and Frameworks 3.1.1, back up your existing Workbench and application configuration, delete your current installation of Tools and Frameworks, and install version 3.1.1.

Backing up the Workbench configuration files

Oracle Endeca Workbench uses several configuration files located in `%ENDECA_TOOLS_CONF%\conf` (on Windows) or `$ENDECA_TOOLS_CONF/conf` (on UNIX) to customize the behavior of various aspects of Workbench.

These files store Workbench configuration, user authentication configuration, and definitions of the menus and extensions in Workbench. If you have manually modified any of the following files from their default state, you should copy them to a backup location:

File name	Description
`Login.conf`	Configuration for user authentication using LDAP
`webstudio.properties`	Miscellaneous configuration parameters for Workbench
`webstudio.log4j.properties`	Configuration for the Workbench system log and audit log
`ws-extensions.xml`	Definitions of Workbench extensions
`ws-mainMenu.xml`	Definitions of the Workbench navigation menu and launch page
`ws-roles.xml`	This file is deprecated in the 3.1.1 release. User configuration for Workbench is managed in the Endeca Configuration Repository.

File name	Description
	For details, see the *Workbench Administrator's Guide*.

Backing up the application state

The `webstudiostore` and `emanager` directories contain information about your application state, including user and permission settings, preview application settings, content XML, and resource metadata.

To back up the application state directories:

1. Stop the Endeca Tools Service.
2. Copy the `webstudiostore` directory and its subdirectories from `%ENDECA_TOOLS_CONF%\state\` (on Windows) or `$ENDECA_TOOLS_CONF/state/` (on UNIX) to another location.

   This directory contains information such as users and permissions, as well as preview application settings.
3. Copy the `emanager` directory and its subdirectories from `%ENDECA_TOOLS_CONF%\state\` (on Windows) or `$ENDECA_TOOLS_CONF/state/` (on UNIX) to another location.

   This directory contains resource metadata, state information, and content XML.
4. Start the Endeca Tools Service.

Backing up application content from the Endeca Configuration Repository

You back up application configuration in the Endeca Configuration Repository using the `export_site` script provided with the Deployment Template.

The script exports application configuration in a format that can be re-imported to the Endeca Configuration Repository. The script connects to the Endeca Configuration Repository instance for the current application based on the configuration in `AppConfig.xml`.

Important: To guarantee consistent data, ensure that no baseline or partial updates are running during the backup process.

To back up application configuration in the Endeca Configuration Repository:

1. Navigate to the `control` directory of your deployed application, for example, `C:\Endeca\apps\discover\control`.
2. Run the `export_site` script, passing in an optional name for the export file, as in the following examples:

   On Windows:

   ```
   export_site.bat ..\ECR-backups\20121221.xml
   ```

   On UNIX:

   ```
   ./export_site.sh ../ECR-backups/20111221.xml
   ```

If no file name is provided, it defaults to a file named according to the pattern `<site-name>-DD-MM-YYYY.xml` in the working directory.

Upgrading from Tools and Frameworks 3.1.0 to 3.1.1

Before installing Tools and Frameworks 3.1.1, you must back up your existing Workbench and application configuration.

To upgrade from Tools and Frameworks 3.1.0 to 3.1.1:

1. Uninstall Tools and Frameworks 3.1.0. See the *Oracle Endeca Commerce Tools and Frameworks Installation Guide* and perform the tasks in "Uninstalling Oracle Endeca Tools and Frameworks".
2. Install Tools and Frameworks 3.1.1. See the *Oracle Endeca Commerce Tools and Frameworks Installation Guide* and perform the tasks in "Installing Oracle Endeca Tools and Frameworks".

Once you have upgraded your installation of Tools and Frameworks, you can migrate the applications you previously backed up.

About migrating an Endeca application from 3.1.0 to 3.1.1

To migrate an Endeca application from 3.1.0 to 3.1.1, you must restore the application configuration from backups, update the application context files and JAR files, and publish the changes to the MDEX Engine and Endeca Configuration Repository.

Restoring a backup of the Workbench configuration files

You restore your Workbench configuration directory, `%ENDECA_TOOLS_CONF%\conf`, by merging in changes from your backup files into the new installation.

To restore a backup of the Workbench configuration files:

1. Stop the Endeca Tools Service.
2. Open your configuration backup files.
3. Manually merge any configuration changes into the files located at `%ENDECA_TOOLS_CONF%\conf` (on Windows) or `$ENDECA_TOOLS_CONF/conf` (on UNIX).

   Important: The syntax for the `Login.conf` file and tool IDs in `ws-mainMenu.xml` have changed for this release. See the *Workbench Administrator's Guide* for the current syntax.

4. Save and close the files.
5. Start the Endeca Tools Service.

Restoring a backup of the application state

You restore backups of the application state directories, `webstudiostore` and `emanager`, by copying backups into the new installation.

To restore backups of the application state directories:

1. Stop the Endeca Tools Service.
2. Delete the `webstudiostore` directory from `%ENDECA_TOOLS_CONF%\state\` (on Windows) or `$ENDECA_TOOLS_CONF/state/` (on UNIX).
3. Copy the backup of the `webstudiostore` directory, including all its subdirectories, to `%ENDECA_TOOLS_CONF%\state` (on Windows) or `$ENDECA_TOOLS_CONF/state/` (on UNIX).
4. Delete the `emanager` directory from `%ENDECA_TOOLS_CONF%\state\` (on Windows) or `$ENDECA_TOOLS_CONF/state/` (on UNIX).
5. Copy the backup of the `emanager` directory and its subdirectories to `%ENDECA_TOOLS_CONF%\state\` (on Windows) or `$ENDECA_TOOLS_CONF/state/` (on UNIX).
6. Start the Endeca Tools Service.

Restoring a backup of application configuration to the Endeca Configuration Repository

You restore a backup of your application configuration to an instance of the Endeca Configuration Repository by using the `import_site` script provided with the Deployment Template.

The script takes a file created by a previous export and imports its content to the Endeca Configuration Repository. The script connects to the Endeca Configuration Repository instance for the current application based on the configuration in `AppConfig.xml`.

Important: To guarantee consistent data, ensure that no baseline or partial updates are running during the backup process.

To restore a backup of your application configuration in the Endeca Configuration Repository:

1. Navigate to the `control` directory of your deployed application, for example, `C:\Endeca\apps\Discover\control`.
2. Run the `import_site` script, passing in the file name of the backup, as in the following examples:

   On Windows:

   ```
   import_site.bat discover-21-12-2012.xml
   ```

   On UNIX:

   ```
   ./import_site.sh discover-21-12-2012.xml
   ```

Before running the Deployment Template scripts to publish content back to the repository, you must upgrade your application to version 3.1.1 by adding the necessary repository structure. See "Updating a 3.1.0 Endeca application to 3.1.1."

Updating Endeca application JAR files to version 3.1.1

After restoring your content, you must update your application with 3.1.1 JAR files.

To update Endeca application JAR files:

1. Navigate to the `%ENDECA_TOOLS_ROOT%\3.1.1\deployment_template\app-templates\common\config\lib\java` folder (on Windows) or `$ENDECA_TOOLS_ROOT/3.1.1/deployment_template/app-templates/common/config/lib/java` (on UNIX).

2. Copy the following files:
   - `commons-io-1.4.jar`
   - `commons-lang-2.4.jar`
   - `eacComponents-3.1.1.jar`
   - `eacToolkit-3.1.1.jar`
3. Navigate to the `config\lib\java` folder of your application, for example:
   `C:\Endeca\apps\Discover\config\lib\java`
4. Paste the JAR files that you copied into the folder.
5. Delete the previous versions of the JAR files:
   - `eacComponents-3.1.0.jar`
   - `eacToolkit-3.1.0.jar`

Updating a 3.1.0 Endeca application to 3.1.1

Before updating your application, ensure you have restored the Workbench configuration, application state, and Endeca Configuration Repository content from your backups

To update an Endeca application from 3.1.0 to 3.1.1:

1. Navigate to the `migration\workbench\3.1.0-3.1.1\bin` directory of your Tools and Frameworks installation.
   By default, this is `C:\Endeca\ToolsAndFrameworks\3.1.1\migration\workbench\3.1.0-3.1.1\bin` on Windows, or `usr/local/endeca/ToolsAndFrameworks/3.1.1/migration/workbench/3.1.0-3.1.1/bin` on UNIX.
2. Copy the `create_redirects_node` script to the `control` directory of your deployed application.
3. Replace the `endeca_assembler-3.1.0.jar` file in your application with the `%ENDECA_TOOLS_ROOT%\assembler\lib\endeca_assembler-3.1.1.jar` file.
4. If you configure the `MdexContentRequestBroker` inline in your application, update the Assembler context file to configure it as a Java bean.
   For the Discover Electronics reference application, add the bean below:

```
<bean id="contentRequestBroker" scope="request"
 class="com.endeca.infront.content.MdexContentRequestBroker">
    <constructor-arg ref="mdexResource"/>
    <constructor-arg ref="${user.state.ref}"/>
    <constructor-arg ref="navigationState"/>
    <constructor-arg ref="${preview.enabled}"/>
</bean>
```

5. Modify any beans that rely on the `MdexContentRequestBroker` to reference the bean you created in the previous step.
   For Discover Electronics, modify the `CartridgeHandler_ContentSlot` bean as follows:

```
<bean id="CartridgeHandler_ContentSlot"
 class="com.endeca.infront.cartridge.ContentSlotHandler">
 scope="prototype">
    <property name="contentBroker" ref="contentRequestBroker"/>
</bean>
```

6. Restart the Web server running your application.
7. Navigate to the `<app dir>\control` directory.
8. Run the `create_redirects_node` script you copied over in Step 2.
9. Run the `load_baseline_test_data` script.
10. Run the `baseline_update` script to publish updated information to the MDEX Engine.
11. Run the `set_templates` script to push updated templates to the repository.
12. Run the `promote_content` script.

Repeat these steps for each application you wish to update.

Chapter 4

Migrating an Endeca Application from 2.1.2 to 3.1.1

About migrating an Endeca application from 2.1.2 to 3.1.1

The migration process consists of the following high-level steps. Each step is fully described in this chapter.

1. Deploy an empty destination application.

   The empty destination application provides a storage mechanism for the migrated Workbench configuration and application content.

2. Standardize feature names in the source application.

   Certain feature name syntax was allowed in 2.1.2 that is not allowed in 3.1.1. This 2.1.2 syntax allowed names with special characters, spaces, etc. If your 2.1.2 features contain such names, you must modify the names in the source application to conform to the name requirements in 3.1.1.

3. Migrate Deployment Template customizations from the source application to the destination application.
4. Configure the `migrate_workbench` script by specifying source and destination application names and directories in the `config.properties` file.
5. Migrate Workbench configuration by running the `migrate_workbench` script.

   The script converts Workbench configuration from 2.1.2 format to the 3.1.1 format and stores the new configuration as files on disk.

6. Modify your application content to a format compatible with the Content Import Utility.

   The utility is called by `initialize_services` to convert application content and import it into the Endeca Configuration Repository.

7. Upload the new Workbench configuration in 3.1.1 format and upload the application content to the Endeca Configuration Repository by running the `initialize_services` script.
8. Verify the Workbench configuration and application content in Workbench 3.1.1.

Upgrading from Endeca IAP 2.1.2 to Tools and Frameworks 3.1.1

This topic describes how to back up a source application from 2.1.2, uninstall the Endeca IAP 2.1.2 software, and install Tools and Frameworks 3.1.1.

To upgrade from Endeca IAP 2.1.2 to Tools and Frameworks 3.1.1:

1. Back up the Endeca IAP 2.1.2 source application. To do this, perform the following sub-tasks:
   a) Locate the source application in the `Endeca\Apps` directory (on Windows).
   For example, `C:\Endeca\Apps\Discover`
   b) Copy the entire source application to a temporary directory that is outside the Endeca installation directory.
   For example, you might copy it to `C:\temp\MigrationSrcApp\Discover`.
   c) Open a command prompt window and remove the source application from the EAC Central Server by running the `eaccmd` utility and the `remove-app` command.
   For example, `eaccmd remove-app --force --app Discover`
   d) Delete the source application's directory from `Endeca\Apps`.
2. Uninstall Endeca IAP 2.1.2 and all of its components. To do this, perform the following sub-tasks:
   a) Uninstall Oracle Endeca Guided Search 2.1.2 (Workbench with Rule Manager) or Oracle Endeca Experience Manager 2.1.2 (Workbench with Page Builder).
   For either installation, see "Uninstalling Oracle Endeca Workbench" in the *Oracle Endeca Workbench Installation Guide* version 2.1.2 available on the Oracle Technology Network.
   b) Uninstall Oracle Endeca Content Assembler API (Java or .NET) 2.1.2.
   For details, see "Uninstalling the Content Assembler API" in the *Oracle Endeca Workbench Installation Guide* version 2.1.2 available on the Oracle Technology Network.
   c) Uninstall URL Optimization API 2.1 (Java or .NET).
   There is no uninstall program. Simply delete the `SEM` directory from the Endeca installation directory.
   For example, in a default installation on Windows, this directory is in
   ```
   C:\Endeca\SEM\URL Optimization APIs
   ```
   d) Uninstall Oracle Endeca Deployment Template 3.2.2.
   There is no uninstall program. Simply delete the `deploymentTemplate-[VERSION]` directory from the Endeca installation directory.
   For example, in a default installation on Windows, this directory is in
   ```
   C:\Endeca\Solutions\deploymentTemplate-3.2.2
   ```
3. Install Tools and Frameworks 3.1.1.
   For details, see the *Oracle Endeca Commerce Tools and Frameworks Installation Guide* and perform the tasks in "Installing Oracle Endeca Tools and Frameworks".

About the source application and the destination application

The migration process relies on a source application and a destination application. The *source application* is the previous version of the Endeca application version that you are migrating from.

Depending on your migration path, the source application is either a 2.1.2 application or a 3.1.0 application. The *destination application* is the 3.1.1 version that you want to migrate to.

Deploying an empty destination application

You create an empty destination application by running the Deployment Template and specifying a deployment descriptor file that is customized for migration operations.

To deploy an empty destination application:

1. Start a command prompt (on Windows) or a shell (on UNIX).
2. Navigate to `ToolsAndFrameworks\<version>\deployment_template\bin` or the equivalent path on UNIX.
3. From the `bin` directory, run the `deploy` script with the `--app` flag and an argument that specifies the path to the `deploy.xml` descriptor file that has been customized for migration operations. For example:

   ```
   C:\Endeca\ToolsAndFrameworks\3.1.1\deployment_template\bin>deploy
   --app C:\Endeca\ToolsAndFrameworks\3.1.1\migration\workbench\2.1.2-
   3.1.1\migrationApp\deploy.xml
   ```

4. In general, you can accept the default values during the deployment process unless your environment requires different port values. You can also use the same application name as the source application.

   If you need details about running the `deploy` script, see the *Tools and Frameworks Deployment Template Usage Guide*.

After deployment, the destination application is available to store output from the `migrate_workbench` script. Go on to *Migrating Workbench configuration* on page 30.

Standardizing feature names in the source application

Certain feature name syntax was allowed in 2.1.2 that is not allowed in 3.1.1. The 2.1.2 syntax allowed names with special characters, spaces, etc. If your 2.1.2 features contain such names, you must modify the names in the source application to conform to the name requirements in 3.1.1.

The name requirements are as follows:

- If you are running IAP 2.1.2 on a Windows machine, keep in mind that the Windows operating system is case-insensitive. You must uniquely name individual rules within Rule Manager without relying on case to distinguish them.
- The special characters: `/ \ : [ ] | * ? " < >` are not permitted in feature names. If any of the special characters are included in User Profiles, Keyword redirects, Rule names, Rule Group names, Zone names, or Style names, then the `migrate_workbench` script does not convert the feature and logs a warning.
- Style names cannot contain spaces or the word `and`. If they are included in a Style name, then the script does not covert the Style, *or any of the rules that use the style*, and the script logs a warning.

Migrating Deployment Template customizations

This topic provides high-level instructions about how to migrate an application from the standalone Deployment Template 3.2.2 to the current version of the Tools and Frameworks Deployment Template 3.1.1.

Note: Remember, the Deployment Template included with Tools and Frameworks has the same version number as Tools and Frameworks. For example, in this release, both are version 3.1.1. The older standalone installation, Deployment Template 3.2.2, is still versioned separately.

Due to the flexible nature of the Deployment Template and the opportunities for customization, specifying a comprehensive migration path from a previous version is not possible. It is the job of Deployment Template users to know how they have customized their deployment so that the appropriate modifications can be retained.

However, it is possible to follow high-level steps that guide the migration path. Above all, you must be aware of the customizations that you made to the previous Deployment Template files so that you can port them to the newer version.

To migrate Deployment Template customizations:

1. Port the changes that you made to `AppConfig.xml` in the source application over to `AppConfig.xml` and its supporting configuration files in the destination application.

   Remember, in Tools and Frameworks 3.1.0 and later, the Deployment Template provides a single `AppConfig.xml` file that contains pointers to other files that define distinct parts of an application, separate scripts from component provisioning, and are used for other purposes.
2. Replace the Deployment Template 3.1.1 pipeline with your custom pipeline.
3. Place your input source data in the appropriate directory.
4. Place any custom Java packages into the Deployment Template `lib/java` directory.
5. Run the `initialize_services` script, load the source data, and run a baseline update.

Configuring the migrate_workbench script

Before running the `migrate_workbench` script, you may need to modify its `config.properties` file if your source or destination applications have customized directory structure or custom application files. In this case, specify the paths to the custom directories and files. You can also indicate the application files that you want to exclude or include in the migration process.

However, if your source and destination applications are based on the default Deployment Template, and you have not made any customizations, you should not need to modify the default directories or files specified in `config.properties`.

The settings in `config.properties` are defined as follows:

Setting	Description
`migrationPath`	Specifies the version numbers of the source application to migration from and the destination application to migration to. The default value is `2.1.2-3.1.1`. **Note:** In this release, do not modify this value. The script supports migrating from 2.1.2 to 3.1.1. (This setting is

Setting	Description
	intended for future use when other migration paths are supported.)
`source.sourceDataDir`	Specifies the data directory of source application. This is the directory where post-Forge state is stored including post-Forge dimensions and dynamic business rules. If your application is customized to store Forge output in a different directory, specify the directory in this setting. The default value is `da¬ta/forge_output`.
`dest.destIfcrDir`	Specifies the IFCR directory of destination application. The default value is `config/ifcr` and then the script takes the `dest` value taken from the `--dest-app-dir` command line parameter.
`pageTemplatesDir`	Specifies the cartridge templates directory of destination application. If your application is customized to store templates in a different directory, specify the directory in this setting. The default value is `config/cartridge_templates`.
`source.pipelineDir`	Specifies the pipeline directory of *source* application. This is the directory where the instance configuration XML files are stored. If your application is customized to store instance configuration XML in a different directory, specify the directory in this setting. The default value is `config/pipeline` within the source application.
`dest.pipelineDir`	Specifies the pipeline directory of *destination* application. This is the directory where the instance configuration XML files are stored. If your application is customized to store instance configuration XML in a different directory, specify the directory in this setting. The `migrate_workbench` script copies the XML files from the value specified in `source.pipelineDir` to the value specified in `dest.pipelineDir`. The default value of `dest.pipelineDir` is `config/pipeline` within the destination application.
`MerchRuleMigrator.enable¬GroupMigration`	Specifies whether to convert dynamic business rule groups into folders that display in Rule Manager and Experience Manager. Setting this value to `true` converts rule groups to folders. The default value is `false`. **Note:** In IAP 2.1.2, rules could trigger across rule groups. In 3.1.1, rules can trigger only within a folder. If you want to replicate 2.1.2 behavior, set this value to `false` so rules are grouped into a single folder.
`excludeFiles`	Specifies which files to exclude from being copied over from the source application to the destination application. The `excludeFiles` setting is a comma separated list of file names. You can specify an absolute file name or use wildcards to replace the application prefix in the XML file name.

Setting	Description
includeFiles	Specifies which files to copy from the source application to the destination application. The includeFiles setting overrides any files you specify in the excludeFiles list. For example, the excludeFiles property uses a default pattern of *.merch_rule_group_* to exclude all the dynamic business rules. If you have a custom file to copy named <app name>.merch_rule_group_customized_file.xml, you can add *.merch_rule_group_customized_file.xml into the includeFiles list. The includeFiles setting is a comma separated list of file names. You can specify an absolute file name or use wildcards to replace the application prefix in the XML file name. The default value is an empty list. An empty list includes all files from the source application into the destination application unless the files are otherwise specified in excludeFiles.

To configure the migrate_workbench script:

1. On the file system, navigate to Endeca\ToolsAndFrameworks\migration\workbench\2.1.2-3.1.1.
2. Open config.properties in a text editor.
3. Save and close config.properties.

After modifying the config.properties file, go on to run the script. For details, see *Migrating Workbench configuration* on page 30.

Migrating Workbench configuration

Once the migrate_workbench script is configured, you can run it to convert Workbench configuration from the 2.1.2 format to the 3.1.1 format.

The migrate_workbench script converts the following features:

- Thesaurus entries
- User profiles (formerly known as user segments)
- Keyword redirects
- Rule groups, zones, and styles
- Dynamic business rules created with Rule Manager (but not created with Page Builder)

The script also copies the instance configuration XML files (i.e. the Developer Studio pipeline files) for features that do not require a format change. The pipeline files are moved from the source application to the destination application.

Note: Any existing configuration for these files in the destination application is deleted at the start of the script. Ensure that all Workbench configuration you wish to migrate is correct before

running the script, as you cannot merge configuration by running the script multiple times on different versions of the configuration files.

Before performing this task, ensure you have backed up the source application and have it available to the local file system where you are running the `migrate_workbench` script. For details, see step #1 of *Upgrading from Endeca IAP 2.1.2 to Tools and Frameworks 3.1.1* on page 26.

To migrate Workbench configuration:

1. Open a command prompt window and navigate to `Endeca\ToolsAndFrameworks\<version>\migration\workbench\2.1.2-3.1.1\bin`.
2. Run the `migrate_workbench` script and specify the following options:
   - `--source-app-dir` - the path to the source application directory you want to migrate from.
   - `--source-app-name` - the name of source application that you want to migrate from.
   - `--dest-app-dir` - the path to the destination application you want to migrate to.
   - `--dest-app-name` - the name of destination application that you want to migrate to.
   - `--config` - the path to the `config.properties` that you modified in step 2.

   For example:

```
C:\Endeca\ToolsAndFrameworks\3.1.1\migration\workbench\2.1.2-
3.1.1\bin>migrate_workbench.bat
--source-app-dir C:\temp\MigrationSrcApp\Discover
--source-app-name Discover
--dest-app-dir C:\Endeca\Apps\Discover
--dest-app-name Discover
--config C:\Endeca\ToolsAndFrameworks\3.1.1\migration\workbench\2.1.2-
3.1.1\config.properties
```

The script converts the Workbench configuration for each application feature in 2.1.2 format and stores the 3.1.1 configuration in JSON files on disk. After conversion, the following 2.1.2 features map to their new 3.1.1 feature equivalents:

- Dynamic business rules become content items
- Rule groups become folders
- Zones become collections (with a content type of `LegacyRule`)

For information about script logging, see *About logging for the migrate_workbench script* on page 50.

After running the script, go on to *About importing 2.1.2 application content* on page 31.

About importing 2.1.2 application content

Oracle Endeca Tools and Frameworks 3.1.1 includes a Content Import Utility that you can use to import application content to the Endeca Configuration Repository.

The utility can import the following types of application content:

- Content items — a set of trigger conditions with associated content XML consisting of property/value mappings.
- Content collections — groups of content items that are evaluated against each other at query time.
- Content collection folders — folders used to organize content collections in Experience Manager or Rule Manager.

Data requirements

Before you can import data to the ECR, you must ensure that it meets the input format requirements described in the following sections.

Data in the Endeca Configuration Repository is stored in a hierarchy that includes nested content and accompanying property files and XML.

The Content Import Utility supports three input types:

- Directories — These specify the desired hierarchy in the content repository.
- JSON files — These define the properties of the directories that you import.
- Non-JSON files — These contain your application data, such as content XML.

Content collection folder requirements

To import content collection folders into the ECR, you must organize the data as described in this section.

File Structure

Folders are purely structural, and are meant as an organizational tool to make application content easier to administer in Experience Manager. Because their type is typically set via the default child type of the folder's parent, folders do not require a nested JSON file to specify properties.

On disk, a set of content collection folders might be organized as follows:

- MyFolderA *(content collection folder)*
  - MyCollectionA *(content collection)*
    - ...
    - ...
  - MyCollectionB *(content collection)*
- MyFolderB *(content collection folder)*
  - ...
  - ...

Properties and Children

A content collection folder only has one property, its `ecr:type`. This value is determined by the default child type of the folder's parent.

`ecr:type`	Child Types
`content-collec¬ tion-folder`	`content-collection-folder` (default) , `content-collection`

Content item and content collection requirements

To import content collections and content items to the ECR, you must organize the data as described in this section.

File Structure

Content items and content collections are represented by a directory structure in the repository. Each directory requires an accompanying JSON file that specifies the properties for the content item or collection. Additionally, a directory that represents a content item must include an XML file with the content XML for that item.

This approach has the advantage of keeping all data for a given content item or collection in a single directory.

Both the JSON file and any XML files for a given entity are prefixed with an underscore (_) character. On disk, this might look like the following:

- MyCollection *(content collection)*
  - _MyCollection.json *(content collection property file)*
  - MyItem *(content item)*
    - _MyItem.json *(content item property file)*
    - _MyItem.xml *(content item content XML)*

Properties and Children

`ecr:type`	Child Types	Properties
`content-collec¬tion`	`content-item`	• `contentType` — A string that specifies the content type that the collection contains. • `ruleLimit` — An integer that contains the evaluation limit for the collection.
`content-item`	None	For content: • `priority` — An integer that determines the priority of the content item relative to other content items in the same collection. • `workflowState` — The state of the content item, either "ACTIVE" or "INACTIVE." • `startTime` — Optional. The date and time that the content item becomes active, using the following format: `yyyy-mm-ddThh:mm` (using a 24 hour clock). • `endTime` — Optional. The date and time that the content item becomes inactive. • A nested `_<Item_Name>.xml` file that contains content XML. • Nested triggers, containing the properties below: For nested triggers: • `searchTerms` — A string that specifies the search term that trigger the content item. If you specify a value for this property, you must set a `matchmode`. • `dvalIDs` — A string array that specifies the combination of selected dimension values (which translate to the end user's Guided Navigation state) that trigger the content item. • `exactLocation` — A Boolean that specifies whether the content item only triggers for the specified navigation state.

ecr:type	Child Types	Properties

		• matchmode — The match mode, from one of the following:
		  • "MATCHPHRASE"
		  • "MATCHEXACT"
		  • "MATCHALL"
		
		If you specify a value for this property, you must set searchTerms.

Memory Requirements

Oracle recommends allocating 4 GB of heap space for every 10,000 content items that you are importing. You can reduce heap space for normal operation after you have finished importing content.

Example

Consider the following subset of data from the Discover Electronics reference application:

- Content
  - Shared *(folder)*
    - Guided Navigation *(content collection)*
      - Category - Cameras *(content item)*

On disk, this data is organized as follows:

- Content\Shared\Guided Navigation_Guided Navigation.json

  This is the property file for the **Guided Navigation** content collection.:

```
{
    "ecr:type": "content-collection",
    "contentType": "SecondaryContent",
    "ruleLimit": 1
}
```

- Content\Shared\Guided Navigation\Category - Cameras

  This directory represents the **Category - Cameras** content item.

- Content\Shared\Guided Navigation\Category - Cameras_Category - Cameras.json

  This file specifies the properties of the **Category - Cameras** content item:

```
{
    "ecr:type": "content-item",
    "workflowState": "ACTIVE",
    "priority": 10,
    "triggers": [
        {
            "exactLocation": false,
            "dvalIDs": ["101022"]
        }
    ]
}
```

- Content\Shared\Guided Navigation\Category - Cameras_Category - Cameras.xml

This is the content XML file for the **Category - Cameras** content item:

```
<ContentItem type="SecondaryContent"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://endeca.com/schema/content/2008">
  <TemplateId>GuidedNavigation</TemplateId>
  <Name>Guided Navigation</Name>
  <Property name="navigation">
    <ContentItemList type="Navigation">
      <ContentItem type="Navigation">
        <TemplateId>RefinementMenu</TemplateId>
        <Name>Price Range</Name>
        <Property name="dimensionName">
          <String>product.price_range</String>
        </Property>
        ...
      </ContentItem>
    </ContentItemList>
  </Property>
</ContentItem>
```

About creating content XML

The migration utilities included with Oracle Endeca Tools and Frameworks 3.1.1 do not include functionality for generating content XML files from application data. If you are migrating from version 2.1.2, you must create any content XML you wish to import to the repository.

Experience Manager templates in 3.x use `Item`, `List`, and `Property` elements defined in the Endeca xavia namespace. Your content XML must use this namespace and conform to the associated schema.

The exact structure of content XML depends on the templates in an application. Creating this content XML consists of the following general steps:

1. For each content item, start by creating a copy of the <`ContentItem`> element from the corresponding template file.
2. Add the necessary namespace declarations and attributes to the <`ContentItem`>.
3. Refer to the *Assembler API Reference (Javadoc)* and set the required member variables of the associated Java class. These should match up with the Java Bean referenceable members of the configuration class in the `com.endeca.infront.cartridge` package.

   Include variables as <`Item`>, <`List`>, or <`Property`> elements with nested values.

Creating content XML for a content item

To create content XML, create a `content.xml` file in the Endeca xavia namespace that includes the necessary properties for the content item you are configuring.

When mapping a cartridge configuration Java class to <`Item`>, <`List`>, or <`Property`> elements in the xavia namespace, keep the following in mind:

- Java class instances become an <`Item`> element with a `class` attribute that specifies the Java class.
- Member variables that are primitive types are nested within the <`Item`> element as <`Property`> or <`List`> elements, with nested <`String`> or <`Boolean`> elements specifying values.

- Member variables that are instances of another Java class are nested within the <Item> element as <Item> elements of their own, which follow the same rules for setting member variables of those classes.

To create content XML for a content item:

1. Open the the 3.1.1 cartridge template that corresponds to the content item you wish to define.

   Consider the "Top Rated Products" content item in the Discover Electronics Reference Application. The associated Horizontal Record Spotlight cartridge template is available within the deployed application at <app dir>\config\cartridge_templates\MainContent-HorizontalRecordSpotlight.xml:

```
<ContentTemplate
  xmlns="http://endeca.com/schema/content-template/2008"
  xmlns:editors="editors"
  xmlns:xavia="http://endeca.com/schema/xavia/2010"
  type="MainContent"
  id="HorizontalRecordSpotlight">
    <Description>Displays selected records horizontally in the main content
 area.</Description>
    <ThumbnailUrl>/ifcr/tools/xmgr/img/template_thumbnails/Main_Horizon¬
talRecordSpotlight.png</ThumbnailUrl>
    <ContentItem>
        <Name>Spotlight Records</Name>
        <Property name="title">
            <String>Featured Cameras</String>
        </Property>
        <Property name="maxNumRecords">
            <String>10</String>
        </Property>
        <Property name="recordSelection">
            <xavia:Item class="com.endeca.infront.cartridge.RecordSpot¬
lightSelection" />
        </Property>
        <Property name="showSeeAllLink">
          <Boolean>false</Boolean>
        </Property>
        <Property name="seeAllLinkText">
          <String />
        </Property>
    </ContentItem>

    <EditorPanel>
        <!-- cartridge editor elements removed from this example -->
    </EditorPanel>
</ContentTemplate>
```

2. Copy the outermost <ContentItem> from the template to a new file, content.xml.

   This is the root tag of your new content XML document. For the example above, this results in the following:

```
<ContentItem>
    <Name>Spotlight Records</Name>
    <Property name="title">
        <String>Featured Cameras</String>
    </Property>
    <Property name="maxNumRecords">
        <String>10</String>
    </Property>
    <Property name="recordSelection">
```

```
        <xavia:Item class="com.endeca.infront.cartridge.RecordSpotlight¬
Selection" />
    </Property>
    <Property name="showSeeAllLink">
      <Boolean>false</Boolean>
    </Property>
    <Property name="seeAllLinkText">
      <String />
    </Property>
</ContentItem>
```

3. Specify the XML and Endeca namespaces as attributes of the content item.

   For example:

```
<ContentItem
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://endeca.com/schema/content/2010">
    <Name>Spotlight Records</Name>
    <!-- additional elements removed from this example -->
</ContentItem>
```

4. Copy the <ContentItem> type attribute from the root <ContentTemplate> element in the cartridge template definition.

   For the example above, this results in the following:

```
<ContentItem type="MainContent"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://endeca.com/schema/content/2010">
    <Name>Spotlight Records</Name>
    <!-- additional elements removed from this example -->
</ContentItem>
```

5. Add a nested <TemplateId> element and set the value to the id attribute of the cartridge template.

   For example:

```
<ContentItem type="MainContent"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://endeca.com/schema/content/2010">
    <TemplateId>HorizontalRecordSpotlight</TemplateId>
    <Name>Spotlight Records</Name>
    <!-- additional elements removed from this example -->
</ContentItem>
```

6. Specify a unique value for the <Name> element.

   This is the display name of the content item in Experience Manager:

```
<ContentItem type="MainContent"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://endeca.com/schema/content/2010">
    <TemplateId>HorizontalRecordSpotlight</TemplateId>
    <Name>Top Rated Products</Name>
    <!-- additional elements -->
</ContentItem>
```

7. Populate the remaining <Property> elements by referring to the *Assembler API Reference (Javadoc)* for the specified classes.

Example: "Top Rated Products," continued

Continuing the example of the "Top Rated Products" content item, the content XML requires values for the following properties:

```
<ContentItem type="MainContent"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://endeca.com/schema/content/2010">
     <TemplateId>HorizontalRecordSpotlight</TemplateId>
    <Name>Top Rated Products</Name>
    <Property name="title">
        <String>Featured Cameras</String>
    </Property>
    <Property name="maxNumRecords">
        <String>10</String>
    </Property>
    <Property name="recordSelection">
       <xavia:Item class="com.endeca.infront.cartridge.RecordSpotlightSe¬
lection" />
    </Property>
    <Property name="showSeeAllLink">
      <Boolean>false</Boolean>
    </Property>
    <Property name="seeAllLinkText">
      <String/>
    </Property>
</ContentItem>
```

The primitive types for the `title`, `maxNumRecords`, `showSeeAllLink`, and `seeAllLinkText` `<Property>` elements can be left as template defaults, or set as necessary.

Setting the nested value of the `<Property name="recordSelection">` is more complicated. Nested `<xavia:Item>` Elements correspond to cartridge configuration classes in the Assembler.

1. Start by referring to the documentation for the `com.endeca.infront.cartridge.RecordSpot¬lightSelection` class in the *Assembler API Reference (Javadoc)*.

   The class includes the following properties:

   - `FilterState filterState` — The filter state to apply to the data set in order to return relevant records.
   - `boolean augment` — Whether the specified filter state should be combined with the current navigation state when determining which records to return.
   - `int recordLimit` — The maximum number of records to include that match the above filter state.
   - `SortOption sortOption` — The sorting method to apply to the records.

2. Determine which properties to specify on the content item.

   Sorting, for example, is frequently left to the end user. In this case, no sort value is specified, so the records are sorted according to the default Sort, or whichever Sort the end user manually selects in the application.

3. Add the properties as nested `<Property>` elements.

   Because `FilterState` is not a primitive type, it must be included as an `<Item>` element with nested `<Property>` elements:

   ```
   <Property name="recordSelection">
     <Item class="com.endeca.infront.cartridge.RecordSpotlightSelection"
       xmlns="http://endeca.com/schema/xavia/2010">
       <Property name="augment">true</Property>
   ```

```
    <Property name="filterState">
      <Item class="com.endeca.infront.navigation.model.FilterState">
      </Item>
    </Property>
    <Property name="recordLimit">8</Property>
  </Item>
</Property>
```

4. Repeat Steps 1-2, this time for the `com.endeca.infront.navigation.model.FilterState` class.

   The relevant properties are:

   - `List<String> navigationFilters` — A List of dimension value IDs that define the selected navigation state.
   - `List<RangeFilter> rangeFilters` — A List of `RangeFilter` objects, each of which defines a record property and valid upper and lower bounds for its value.
   - `List<String> recordFilters` — A List of record IDs that define a selection of specified records.
   - `List<SearchFilter> searchFilters` — A List of `SearchFilter` objects, each of which defines a search term, search key, and match mode.

   Recall that this content item is a dynamic record spotlight for highly rated records; however, for navigability in the application, records are tagged with dimension values for various review ranges during back-end ingest and processing. For this reason, the content item requires a `navigation¬Filters` value (rather than a `rangeFilters` value, as might be expected).

5. Add the property as a nested `<Property>` element with one or more nested `<String>` elements:

```
<Property name="recordSelection">
  <Item class="com.endeca.infront.cartridge.RecordSpotlightSelection"
    xmlns="http://endeca.com/schema/xavia/2010">
    <Property name="augment">true</Property>
    <Property name="filterState">
      <Item class="com.endeca.infront.navigation.model.FilterState">
        <Property name="navigationFilters">
          <List>
            <String>100021</String>
          </List>
        </Property>
      </Item>
    </Property>
    <Property name="recordLimit">8</Property>
  </Item>
</Property>
```

6. Continue to repeat these steps as necessary until all `<Property>` elements have been populated with values.

The final content XML for the "Top Rated Products" content item is shown below:

```
<ContentItem type="MainContent"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://endeca.com/schema/content/2008">
  <TemplateId>HorizontalRecordSpotlight</TemplateId>
  <Name>Top Rated Products</Name>
  <Property name="title">
    <String>Top Rated</String>
  </Property>
  <Property name="maxNumRecords">
    <String>10</String>
```

```
  </Property>
  <Property name="recordSelection">
    <Item class="com.endeca.infront.cartridge.RecordSpotlightSelection"
    xmlns="http://endeca.com/schema/xavia/2010">
      <Property name="augment">true</Property>
      <Property name="filterState">
        <Item class="com.endeca.infront.navigation.model.FilterState">
          <Property name="navigationFilters">
            <List>
              <String>100021</String>
            </List>
          </Property>
        </Item>
      </Property>
      <Property name="recordLimit">8</Property>
    </Item>
  </Property>
  <Property name="showSeeAllLink">
    <Boolean>false</Boolean>
  </Property>
  <Property name="seeAllLinkText">
    <String/>
  </Property>
</ContentItem>
```

Importing Workbench configuration and application content

The `initialize_services` script imports content to the Endeca Configuration Repository. This occurs once, when you initialize an application.

Important: You cannot import content into an Assembler application that is already provisioned in the Endeca Application Controller.

To uploading Workbench configuration and application data to the repository:

1. If your source application processed file-based data extracts, then update the data extract process to copy the extract to the new `<app dir>\data\incoming` directory of the destination application.
2. Copy the content you wish to import to the Endeca Configuration Repository into the `<app dir>\config\import\content` directory of the destination application.
3. Open a command prompt window and navigate to the `control` directory of the destination application.
4. Run the `initialize_services` script.

   This provisions your application in the Endeca Application Controller and uploads your data to the Endeca Configuration Repository.

   Note: This operation may run for up to 20 minutes per 10,000 content items.

5. Run the `load_baseline_test_data` script.
6. Run the `baseline_update` script.
7. Run the `promote_content` script.

Verifying the Workbench configuration and application data

After importing the Workbench configuration and application content, you can verify that the features migrated correctly by viewing them in Workbench.

To verify the Workbench configuration and application data:

1. Verify that the Workbench configuration migrated correctly by doing the following:
   a) Start a Web browser and log in to Workbench 3.1.1.
   b) Select the **Thesaurus** page and check the list of thesaurus entries.
   c) Select the **User Segments** page and check the list of user segments.
   d) Select the **Keyword Redirects** page and check the list of keyword redirects.
2. Verify that the application content imported correctly by doing the following:
   a) Select the **Rule Manager** page or the**Experience Manager** page and check that dynamic business rules are content items.
   b) If you indicated that rule groups should be converted, select the **Rule Manager** page or **Experience Manager** page and check that rule groups are folders.
   c) Select the **Rule Manager** page or **Experience Manager** page and check that zones are collections.

Migrating front-end application code

You must refactor your application to use the 3.1.1 Assembler instead of the Content Assembler API.

The Assembler can run in process as part of a Java application, or it can be deployed as a standalone servlet. For details on these two modes of operation, refer to the *Assembler Application Developer's Guide*.

Migrating an application from the Content Assembler API to the 3.1.1 Assembler consists of the following high level steps:

1. Update the application code to use the new Assembler API, or to invoke the Assembler service.
2. Update the cartridge templates to use the properties in the Assembler.
3. Update the tag handlers and cartridge handlers.
4. Update how the application accesses MDEX Engine query results.
5. Update the application logic for constructing and optimizing URLs.

About migrating an application from the Java Content Assembler API to the Assembler

Query parameters in the Content Assembler API were centralized in the `ENEQuery` class; associated URL parameters existed in the `UrlENEQuery` class. In the 3.x Assembler, query configuration has been modified for the content item-centric configuration model.

Global query parameters are marshaled and set by an instance of the `UrlNavigationStateBuilder`, which is typically configured in the Assembler context file of an application. The class serializes an application end user's actions into navigation state and record state objects used by the various content items to query the MDEX Engine. To customize how user actions translate to a navigation state, you can add your own logic to a subclass of the `UrlNavigationStateBuilder`.

Query configuration for the individual content items themselves has been relocated to the interfaces and cartridge configuration models corresponding to those items. For example, if your application features logic for processing URL parameters in a given cartridge, you should extend the `RequestParamMarshaller` class for that cartridge and include the logic there. To modify URL parameter handling in the Results List cartridge, you would extend the `ResultsListRequestParamMarshaller` class.

To see the mappings between classes, methods, and URL parameters in the Content Assembler API and those in the Assembler, refer to *Mappings between the Content Assembler API and the Assembler* on page 42.

For an overview of the cartridge configuration model, see the chapter "Configuring Front-End Application Features" in the *Assembler Application Developer's Guide*.

About migrating an application from the .NET Content Assembler API to the Assembler

If you are migrating a .NET application that previously used the .NET Content Assembler API, you must modify your application to use the 3.1.1 Assembler as a service. Your must restructure your application query logic to pass in parameters via URL on the inbound request to the Assembler service.

To see the mappings between URL parameters in the Content Assembler API and those in the Assembler, refer to *Mappings between the Content Assembler API and the Assembler* on page 42.

For an overview of the cartridge configuration model, see the chapter "Configuring Front-End Application Features" in the *Assembler Application Developer's Guide*.

Mappings between the Content Assembler API and the Assembler

This is an overview of the mappings between the methods and parameters that enabled features in the Content Assembler API, and the corresponding methods and parameters in the 3.1.1 Assembler.

Query configuration mappings

Global configuration for the features below is typically set in the Assembler context file on the class and property specified in the table.

Feature	CA API URL Param	Assembler URL Param	Global Configuration <class>.<property>	Cartridge Handler(s)
Navigation query	N	N	`--`	`UrlNavigationStateBuilder`
Refinement display in menu	Nrc	Nrmc	`RefinementMenuConfig.refinementsShown`	RefinementMenu
Enable "Show More Refinements" link			`RefinementMenuConfig.showMore`	RefinementMenu
"Show More" dimension IDs			`NavigationContainer.showMoreIds`	NavigationContainer

Feature	CA API URL Param	Assembler URL Param	Global Configuration <class>.<property>	Cartridge Handler(s)
Record details	R	R	`DefaultResult¬sListConfig`	UrlNavigationStateBuilder
Record offset	No	No	`ResultsListCon¬fig.offset`	ResultsList
Records to show per aggregate record	Np		`ResultsListCon¬fig.subRecordsPer¬AggregatRecord`	ResultsList
Record filter	Nr		`Filter¬State.recordFil¬ters`	UrlNavigationStateBuilder
Record boost/bury				
Records per page		Nrpp	`ResultsListCon¬fig.recordsPer¬Page`	ResultsList
Record search key	Ntk	Ntk	`Filter¬State.SearchFil¬ters.key`	ResultsList, DimensionSearchResult
Aggregate record selection	A	A	`--`	UrlNavigationStateBuilder
Aggregate record offset	Nao	Nao	`ResultsListCon¬fig.offset`	ResultsList
Aggregate record rollup key	Nu		`Filter¬State.rollupKey`	UrlNavigationStateBuilder
Why Rank		whyrank	`ResultsListCon¬fig.whyRankEn¬abled`	ResultsList
Why Match	Nx	whymatch	`ResultsListCon¬fig.whyMatchEn¬abled`	ResultsList
Why Precedence Rule Fired		`whyprece¬dencerule¬fired`	`Refinement¬Menu.whyPrece¬denceRuleFired, NavigationContain¬er.whyPrece¬denceRuleFired`	RefinementMenu, NavigationContainer
Range filter	Nf	Nf	`FilterState.range¬Filters`	UrlNavigationStateBuilder
Set preview time	Nmpt	Endeca_Time	`UserState.date`	`--`

Feature	CA API URL Param	Assembler URL Param	Global Configuration <class>.<property>	Cartridge Handler(s)
Relevance ranking Match Mode	Nrm	--	Filter¬ State.SearchFil¬ ters.MatchMode	UrlNavigationStateBuilder
Relevance ranking strategy	Nrr	--	ResultsListCon¬ fig.relRankStrate¬ gy, Dimension¬ SearchResultsCon¬ fig.relRankStrate¬ gy	ResultsList
Relevance ranking search terms	Nrt	Nrt	--	--
Relevance ranking search key	Nrk		--	--
EQL filter	Nrs	Nrs	--	UrlNavigationStateBuilder
Sort key	Ns	Ns	ResultsListCon¬ fig.sortOption, Refinement¬ Menu.sort	ResultsList, RefinementMenu
Sort order	Nso			
Compute phrasings	Ntpc	Ntp	SearchAdjust¬ mentsConfig.phras¬ eSuggestionEn¬ abled	UrlNavigationStateBuilder
Rewrite query with alternate phrasing	Ntpr			
Search terms	Ntt	Ntt	Filter¬ State.SearchFil¬ ters.terms	UrlNavigationStateBuilder
Search mode	Ntx	Ntx	Filter¬ State.SearchFil¬ ters.matchMode	UrlNavigationStateBuilder
"Did You Mean"	Nty	Nty	SearchAdjust¬ mentsConfig.spell¬ SuggestionEnabled	UrlNavigationStateBuilder
Signal dimension search	--	Dy		
Dimension search term	D	Ntt with Dy=1		
Dimension search range filter	Df			

Feature	CA API URL Param	Assembler URL Param	Global Configuration <class>.<property>	Cartridge Handler(s)
Enable dimension search relevance ranking	Dk			
Dimension search scope	Dn	Dn		
Dimension search result offset	Do	Do		
Dimension search dimVal count	Dp	Dp		
Dimension search record filter	Dr	Dr		
Dimension search refinement configuration	Drc	- -		
Dimension search EQL filter	Drs	- -		
Dimension search options	Dx	- -		

Refactored parameters

In addition to the re-mapped parameters listed in the table above, the following parameters have had their functionality rolled into existing parameters in the Oracle Endeca Tools and Frameworks 3.1.1 release:

Feature	Content Assembler API URL parameter	New parameter
Nav search options	`Nx`	`whymatch`, `whyrank`, and `whyprecedencerule¬fired` parameters.
Configure refinements	`Nrc`	`Nrmc`
Dimension refinement boost / bury	`Nrcs`	Dimension refinement boost and bury behavior is configured on the `RefinementMenuConfig` object, but does not include associated URL parameters.
Aggregate record range filter	`Af`	`Nf`
Aggregate record descriptors	`An`	`Nn`
Aggregate record filter	`Ar`	`Nr`
Aggregate record EQL filter	`Ars`	`Nrs`

Feature	Content Assembler API URL parameter	New parameter
Aggregate record sort key	`As`	`Ns`
Aggregate record rollup key	`Au`	`Nu`
Dimension search rollup key	`Du`	`Nu`
Dimension search ID filter	`Di`	Results are already limited to the IDs specified for dimension search

Unsupported features in the 3.1.1 Assembler

The following features in the 2.1.2 Content Assembler for Java and .NET do not have equivalent functionality in the 3.1.1 Assembler:

Feature	Content Assembler API URL parameter
Show disabled refinements	`Ndr`
Specify exposed refinements	`Ne`
Trigger Merchandising Rule filter	`Nmrf`

About template changes

As part of migrating an application, you must update the cartridge templates to use the property types and classes in the 3.1.1 Assembler.

Primitive property types do not require changes, but you must migrate property types with associated classes to the corresponding new types in the Assembler. For example, the `RecordList` property in 2.1.2 corresponded to a list of selected records. In Tools and Frameworks 3.1.1, this functionality is part of the `RecordSpotlightSelection` class. In a cartridge template, this is represented as the following Endeca xavia element:

```
<xavia:Item class="com.endeca.infront.cartridge.RecordSpotlightSelection"
/>
```

The 2.1.2 Content Assembler API also supported the inclusion of pass-through properties in cartridge templates. The 3.1.1 Assembler supports custom cartridge handlers for processing any data you wish to include in a template. To migrate pass-through properties, you must consider the behavior that they drive in your 2.1.2 application, and implement your own cartridge handler code to duplicate that behavior in a 3.1.1 Assembler application. For example, in the 2.1.2 Content Assembler API, the `<NavQuery>` property specifies a hard-coded navigation state that either stands alone or augments the current navigation state. In a 3.1.1 Assembler application, you should treat this as a hard-coded `RecordSpotlightSelection` property within the cartridge template.

To see the mappings between template properties in Tools and Frameworks 2.1.2 and 3.1.1, refer to *Template property mappings between Tools and Frameworks 2.1.2 and 3.1.1* on page 47.

Cartridge editors

In addition to updating the properties in your cartridge templates, you must update the related editors in the `<EditorPanel>`. See the *Assembler Application Developer's Guide* for an overview of mappings between properties and editors in the core cartridges.

Zones and styles

Rule zones and styles are deprecated in the 3.1.1 release, and replaced with the concepts of folders and content collections. Content items are contained within a collection that evaluates all items against each other and returns the top *n* matches, where *n* is the evaluation limit of the content collection or the dynamic slot that contains it. If your application includes logic that is based on restricting trigger conditions to a specific zone, you must replace this logic to use content folders or content collections.

When including a dynamic content slot in a cartridge template, include properties for specifying the content collection that the slot contains, and the evaluation limit for the slot. For example:

```
<ContentItem>
    <Name>Header Content Slot</Name>
    <Property name="contentCollection">
        <String></String>
    </Property>
    <Property name="ruleLimit">
        <String>1</String>
    </Property>
</ContentItem>
```

For additional information on the configuration classes in Tools and Frameworks 3.1.1, see the *Assembler Application Developer's Guide* and the *Assembler API Reference (Javadoc)*.

Template property mappings between Tools and Frameworks 2.1.2 and 3.1.1

Cartridges in your Oracle Endeca implementation may vary from the core cartridges included with the Tools and Frameworks package based on the requirements of your application. The mappings presented below are meant to highlight the conceptual connections between feature implementation in versions 2.1.2 and 3.1.1. You must ensure that the properties included in your cartridge templates map to the appropriate cartridge handlers in your application. For detailed information on cartridge handler configuration, see the documentation for the `com.endeca.infront.cartridge.<cartridge>Con¬fig` classes in the *Assembler API Reference (Javadoc)*.

Reference cartridge mappings

The following table lists the approximate mappings between 2.1.2 and 3.1.1 core cartridges. Use them as a starting point when determining the necessary changes to your own cartridge templates:

2.1.2 Content Assembler Cartridge Template(s)	3.1.1 Cartridge Template(s)
`SidebarItem-Breadcrumbs.xml`	`SecondaryContent-Breadcrumbs.xml`
`MainColumnContent-DimensionSearchResults.xml`	• `MainContent-DimensionSearchResults.xml` • `AutoSuggestPanel-AutoSuggestPanel.xml`
`SidebarItem-GuidedNavigation.xml`	• `SecondaryContent-GuidedNavigation.xml` • `Navigation-RefinementMenu.xml`

2.1.2 Content Assembler Cartridge Template(s)	3.1.1 Cartridge Template(s)
• MainColumnContent-ImageBanner.xml • FullWidthContent-ImageSiteBanner.xml • SidebarItem-ImageBox.xml	• ImageBanner-Image.xml • MainContent-MediaBanner.xml
MainColumnContent-ResultsList.xml	MainContent-ResultsList.xml
MainColumnContent-SearchAdjustments.xml	MainContent-SearchAdjustments.xml
FullWidthContent-SearchBar.xml	HeaderContent-SearchBox.xml
• MainColumnContent-TextBanner.xml • SidebarItem-TextBox.xml	• MainContent-RichTextMain.xml • SecondaryContent-RichTextSecondary.xml
• MainColumnContent-ThreeRecordBanner.xml • MainColumnContent-OneRecordBanner.xml • SidebarItem-ThreeRecordBox.xml	• SecondaryContent-RecordSpotlight.xml • MainContent-HorizontalRecordSpotlight.xml

Content properties

2.1.2 Template Property	3.1.1 Template Property
<Boolean>	Unchanged
<ContentItem>	Unchanged
<ContentItemList>	Unchanged
<NavigationRecords>	The <NavigationRecords> property has been split into multiple properties: • The recordsPerPage attribute exists as its own property with a nested <String> value. • The nested <Sort> element exists as a separate property element with a nested <xavia:Item class="com.endeca.infront.navigation.model.SortOption"> value. • The nested <RelevanceRanking> element exists as a separate property element with a nested <String> value. You should migrate any templates which contain a <NavigationRecords> property to a template based on the Discover MainContent-ResultsList sample template. Use the ResultListHandler to handle any necessary processing logic. For detailed information, see the Javadoc for the com.endeca.infront.cartridge.ResultsListConfig class.

2.1.2 Template Property	3.1.1 Template Property
<NavigationRefine¬ments>	The <NavigationRefinements> property has been split into multiple properties: • The dimension name and ID must exist in the template as <Proper¬ty> elements, each with a nested <String> element. These may optionally be populated with default values. • The Navigation Refinements Selector editor has been replaced with the Dimension Selector editor. . You should migrate any templates which contain a <NavigationRe¬finements> property to a template based on the Discover Naviga¬tion-RefinementMenu sample template. Use the Refinement¬MenuHandler to handle any necessary processing logic. For detailed information, see the Javadoc for the com.endeca.in¬front.cartridge.RefinementMenuConfig class.
<RecordList>	<xavia:Item class="com.endeca.infront.car¬tridge.RecordSpotlightSelection" />
<String>	Unchanged

For details on configuring properties and editors in a cartridge template, see the *Assembler Application Developer's Guide*

Pass-through properties

The mappings for pass-through properties depend on the function of the property in your original application. The following are suggested mappings, and are approximate at best; the best approach for a given application depends on the intended purpose of the property within the application:

2.1.2 Template Property	3.1.1 Template Property
<NavigationResult>	Previously, this tag enabled access to query results from content items nested in the cartridge template. This is no longer necessary, as all content is evaluated and returned in the Assembler response.
<NavQuery>	<xavia:Item class="com.endeca.infront.car¬tridge.RecordSpotlightSelection" />
<RecordQuery>	<xavia:Item class="com.endeca.infront.car¬tridge.RecordSpotlightSelection" />
<Supplement>	A <ContentItem> (specifically, a ContentSlot) with a nested String property that specifies a content collection.
<UrlEneQuery>	You can include the necessary data within a <xavia:Item class="com.endeca.infront.cartridge.model.LinkBuilder" /> element, but you must extend the associated cartridge handler to resolve any hard-coded values included in the cartridge template.

About handling the Assembler response

Whether you are using the Assembler API directly or invoking the Assembler as a service, you must update your application to interpret and render the data in the Assembler response object.

About accessing query results

Applications can no longer access the query result object directly; instead, you must update your application to handle the individual content items returned from the Assembler.

About migrating from tag handlers to cartridge handlers

Unlike tag handlers in the Content Assembler, cartridge handlers in the Assembler are not assigned at an arbitrary level. The Assembler is built around processing queries at the cartridge level, marshaling query parameters and returning results for a single cartridge at a time. Update your application logic accordingly.

Additionally, migrate any tag handler logic to instead use the cartridge handlers that correspond to the new cartridge properties in the application. For additional information, see *Template property mappings between Tools and Frameworks 2.1.2 and 3.1.1* on page 47.

About generating links and optimized URLs

In the Content Assembler API, link generation was largely left to the application developer. The 3.1.1 Assembler generates and interprets SEO-optimized URLs as part of its core functionality, and also handles correct N-value generation when adding or removing refinements from a navigation state.

About logging for the migrate_workbench script

The `migrate_workbench` script logs its migration operations. The log file is stored in the `<app name>\logs\migration` directory of the destination application.

If desired, you can modify the logging level for migration operations by modifying the `log4j.properties` file that is stored in `<install path>\Endeca\migration\workbench`.

Chapter 5

Required Changes

You must make the changes specified in this section if the changes apply to your application.

Required changes in 3.1.1

This section contains changes that are required in version 3.1.1

Workbench content source changes

Assembler API changes

In the Tools and Frameworks 3.1.1 release, the `LiveContentSource` and `AuthoringCon¬tentSource` classes have been deprecated, and their functionality has been incorporated into the `WorkbenchContentSource` class.

The previous constructor for the `WorkbenchContentSource` has been deprecated and replaced with a new constructor that takes no arguments.

The following properties on the `WorkbenchContentSource` class have been deprecated:

- `realPath` and `contextPath` — Oracle recommends using a `clientPort` property to uniquely identify each Assembler application to the `PromotionStatusServlet`.
- `workbenchPort` — This has been replaced with `serverPort`.
- `user`, `password`, and `useSSL` — These properties are no longer required, as the TCP protocol used to connect to the service does not require authentication.

For detailed information on the updated `WorkbenchContentSource` class, see the *Assembler API Reference (Javadoc)*.

Filter State changes

Assembler API changes

The previous version of the `FilterState` constructor, which took multiple arguments, has been deprecated. Instead, use the empty `FilterState()` constructor and set properties using the associated setter methods.

Refinement Menu and Navigation Container configuration

Assembler API changes

To provide more control over the presentation of dimension refinements, the following changes have been made to the Refinement Menu and Navigation Container cartridges:

- The `RefinementMenuConfig` and `NavigationContainerConfig` classes each include a new property, `refinementsShown`. You can set this property to of the following String values:
  - `none` — the refinement menu returns no refinements.
  - `some` — the menu returns `numRefinements` refinements.
  - `all` — the menu returns `maxNumRefinements` refinements. Note that the `NavigationContainerConfig` object cannot use this value for the property.
- The `NavigationContainerConfig` class includes a new property, `refinementsShownByDefault`.
- The `showMoreIds` URL parameter is deprecated. Both cartridges now use the `Nrc` URL parameter to dynamically configure dimension refinement behavior.
- The `RefinementMenuConfig` and `NavigationContainerConfig` classes each include a new property, `useShowMoreIdsParam`.

Note: This property is deprecated. It exists for backwards compatibility.

For detailed information on the updated classes, see the *Assembler API Reference (Javadoc)*.

ResultsListConfig changes

In the Tools and Frameworks 3.1.0 patch release, the `ResultsListConfig` cartridge configuration model has been updated to support Relevance Ranking for search keys, terms, and match modes other than those specified in the search filter:

- The `relrank` property has been renamed to `relRankStrategy`.
- The `getRelRank()` method has been deprecated. Use `getRelRankStrategy()`.
- The `setRelRank()` method has been deprecated. Use `setRelRankStrategy()`.

Additionally, the following properties and methods have been updated:

- The `defaultRecsPerPage` property has been renamed to `recordsPerPage`.
  - The getter and setter methods have been renamed to `getRecordsPerPage()` and `setRecordsPerPage()`.
- The `defaultSort` property has been renamed to `sortOption`.
  - The getter and setter methods have been renamed to `getSortOption()` and `setSortOption()`.
- The `ResultsListConfig()` and `ResultsListConfig(String pType)` constructors have been updated. For a list of default values, see the *Assembler API Reference (Javadoc)*.

Please refer to the *Assembler Application Developer's Guide* and *Assembler API Reference (Javadoc)* for detailed information on these changes.

Related Links

Sorting changes in the Filter State and Select Featured Product dialogs

Editor changes

Microbrowser changes

The microbrowser has been deprecated and replaced with the Select Records dialog.

Cartridge template changes

To support the new Selected Records dialog, the following editors have been deprecated and replaced with new editors:

- The `RecordListEditor` editor is deprecated, and replaced with the `SpotlightSelectionEd¬itor`.
- The `RecordStratificationEditor` editor is deprecated, and replaced with the `Boost¬BuryRecordEditor`.

These updated editors map to the same properties as the editors they replace.

Editor configuration changes

The configuration for the Link Builder editor has been streamlined. The Link Builder formerly supported multiple nested configuration properties that applied to all instances of the editor in an application. This configuration model is deprecated in the current release. The editor now requires a path to a data service:

```
<Editor name="editors:LinkBuilderEditor">
    <EditorConfig resourcePath="/configuration/tools/xmgr/services/ende¬
caBrowserService.json" />
</Editor>
```

For detailed information about editor configuration and template properties, see the *Assembler Application Developer's Guide*.

Required changes in 3.1.0

This section contains changes that are required in version 3.1.0.

The Content Assembler API

The Content Assembler API is deprecated and not compatible with Tools and Frameworks 3.1.0. It has been replaced by the Endeca Assembler.

The Content Assembler API included:

- The Content Assembler API for Java
- The Content Assembler API for the RAD Toolkit for ASP.NET

The RAD Toolkit for ASP.NET

The RAD Toolkit for ASP.NET is deprecated and not compatible with Tools and Frameworks 3.1.0. You can use the Assembler as a service to build a .NET application.

The Deployment Template

Installation changes

In the Tools and Frameworks 3.1.0 release, the Deployment Template has been repackaged as part of Tools and Frameworks. The Deployment Template is no longer a separate installation.

By default, the Deployment Template is installed into `C:\Endeca\ToolsAndFrameworks\<version>\deployment_template` (on Windows) and `usr/local/endeca/ToolsAndFrameworks/<version>/deployment_template` (on UNIX).

Versioning

The Deployment Template included with Tools and Frameworks has the same version number as Tools and Frameworks itself. For example, in this release, both are version 3.1.0. (The older standalone installation, Deployment Template 3.2.2, is still versioned separately.)

Configuration in AppConfig.xml has been split into multiple files

In previous releases of the Deployment Template, most of an application's configuration was described in a single file named `AppConfig.xml`. In Tools and Frameworks 3.1.0, the Deployment Template provides a single `AppConfig.xml` file that contains pointers to other files that define distinct parts of an application, separate scripts from component provisioning, and are used for other purposes.

The full set of application configuration files are as follows:

- `InitialSetup.xml` - Specifies scripts to perform initial setup tasks, such as uploading initial configuration to Workbench.
- `DataIngest.xml` - Specifies data processing scripts, including the baseline update script, partial update script, and the components to perform data processing such as Forge and Dgidx.
- `DgraphDefaults.xml` - Specifies default values that are inherited by all Dgraph components. These values include host IDs, data processing paths, and Dgraph flags.
- `AuthoringDgraphCluster.xml` - Specifies the Dgraphs used in the authoring environment and a script that pushes configuration from Workbench to each Dgraph in the authoring cluster.
- `LiveDgraphCluster.xml` - Specifies the Dgraphs used in the live environment and a script that pushes configuration from Workbench to each Dgraph in the live cluster.
- `WorkbenchConfig.xml` - Specifies the IFCR component, the Workbench Manager component, and a script that promotes content from the authoring environment to the live environment.
- `ReportGeneration.xml` - Specifies the hosts used for logging and report generations, and several scripts that produce log files at different time intervals.

For more information, see the *Tools and Frameworks Deployment Template Usage Guide*.

Removal of CAS WSDL client stubs

The CAS WSDL client stubs have been removed from the Deployment Template packaging.

In previous releases of the Deployment Template, you *replaced* the existing CAS WSDL client stubs in the Deployment Template with client stubs provided in the most recent release of CAS.

If you are using Tools and Frameworks 3.1.0, you *add* the most recent CAS WSDL client stubs to the Deployment Template. For additional information see the *CAS Installation Guide*.

Removal of CAS scripts and the sample CAS pipeline

In the Tools and Frameworks 3.1.0 release, running the `deploy` script no longer creates CAS scripts or a sample CAS pipeline as part of a new application.

In previous releases, the CAS scripts and sample CAS pipeline were designed for crawls that wrote record file output. Both the CAS scripts and sample CAS pipeline configuration to support that process was somewhat complicated.

In new applications, CAS crawls are more easily run as part of `AppConfig.xml` code and do not require additional CAS scripts or the sample CAS pipeline. Therefore, the following scripts and samples are no longer produced for new applications:

- `<installation path>\<deployed app name>\config\cas_crawl_pipeline`
- `<installation path>\<deployed app name>\config\crawl_script_templates`
- `<installation path>\<deployed app name>\config\script\fetchCasCrawlDataConfig.xml`
- `<installation path>\<deployed app name>\control\cas` (This directory included the `baseline_cas_crawl`, `incremental_cas_crawl`, `load_full_cas_crawl_data`, `load_incremental_cas_crawl_data`, and `make_cas_crawl_scripts`.)

If your application runs CAS crawls that write to record file output, you can still run use the CAS scripts that were produced by `make_cas_crawl_scripts`.

Documentation changes

The *Deployment Template Usage Guide* is distributed as part of the Tools and Frameworks documentation set.

Experience Manager editors

The following Experience Manager editors shipped with 2.1.2 but have been removed from 3.1.0:

- <ImagePreview>
- <NavigationRefinementsSelector>
- <NavigationRecordsEditor>
- <RecordSelector>

Experience Manager SDK

Installation changes

In the Tools and Frameworks 3.1.0 release, the Experience Manager Editor SDK has been repackaged as part of Tools and Frameworks. It is no longer a separate installation.

Versioning

The Experience Manager Editor SDK has the same version number as Tools and Frameworks. For example, in this release, both are version 3.1.0.

Experience Manager extensions require newer version of the Flex SDK

In Experience Manager 2.1.2, you compiled extensions using Flex SDK 3.2. In Experience Manager 3.1.x, you must recompile extensions using Flex SDK 4.5. For details, see "Experience Manager Editor Developer's Guide".

Documentation changes

The *Experience Manager Editor Developer's Guide* has been merged into the Tools and Frameworks documentation set.

Endeca Analytics

In the Tools and Frameworks 3.1.0 release, the Endeca Analytics feature, including the Analytics API, is not supported by the Endeca Assembler. Endeca Analytics is still supported in applications that use the Endeca Presentation API.

Chapter 6

Behavioral Changes

This section describes changes that do not require action on your part, but do have an effect on how your Endeca application behaves after you upgrade.

Behavioral changes in 3.1.1

This section describes the changes made in Tools and Frameworks 3.1.1.

Oracle Endeca Workbench

Logging in

In previous releases of Oracle Endeca Workbench, users had to select an application when they logged in to Workbench. In the Tools and Frameworks 3.1.1 release, users only need to enter their user names and passwords to log in to Oracle Endeca Workbench. After logging in, first-time users are presented with a dialog where they can select an application. On subsequent logins, users log in directly to the last application that they accessed. If users have access to multiple applications, they can select another application from the header of an Oracle Endeca Workbench page.

In previous releases of Oracle Endeca Workbench, when unauthenticated users entered the URL for a Workbench page, for example, the URL for the Experience Manager, they were challenged with a login screen and then directed to the Workbench home page. From the home page, users had to navigate back to the Experience Manager. In Tools and Frameworks 3.1.1 release, users are now redirected back to the Workbench page that they initially tried to access after they are authenticated. It is no longer necessary to navigate from the home page. This change allows users to bookmark a page in Oracle Endeca Workbench and login directly that page.

Logging out

In previous releases of Oracle Endeca Workbench, users clicked the **logout** link in the upper right corner of the page to log out of Workbench. Now, users click the down arrow next to their user names in the upper right corner of the page and then click **Logout**.

User Settings

In previous releases of Oracle Endeca Workbench, non-LDAP users could change their passwords using the User Settings tool. In the Tools and Frameworks 3.1.1 release, non-LDAP users can change

their passwords by clicking the down arrow next to their user names in the upper right corner of the page and then clicking **Change Password**.

Unsupported configuration parameters in LDAP login profile

Oracle Endeca Workbench stores LDAP login configuration information in the login profile file, `%ENDECA_TOOLS_CONF%\conf\Login.conf`. In the Tools and Frameworks 3.1.1 release, the following parameters are no longer supported in Workbench:

- `ldapBindAuthentication`
- `passwordAttribute`
- `checkPasswords`

Behavioral changes in 3.1.0

This section describes the changes made in Tools and Frameworks 3.1.0.

Instance Configuration Management and Developer Studio

In the Tools and Frameworks 3.1.0 release, many of the features that you previously configured in Developer Studio are now managed by Oracle Endeca Workbench.

Remember that an *instance configuration* is the set of project files that configure the back-end processes (Forge, Dgidx, Dgraph) of an Endeca implementation. Developer Studio was the interface to these files for many features.

In this release, you no longer configure the following features in Developer Studio:

- **Zones** - Zones are replaced by similar functionality implemented as folders and collections in Experience Manager.
- **Styles** - Styles are replaced by similar functionality implemented as folders and collections in Experience Manager.
- **Rules** - Rules configured in Developer Studio are ignored.
- **Thesaurus entries** - Thesaurus entries are managed by Oracle Endeca Workbench and Experience Manager.
- **Keyword redirects** - Keyword redirects are managed by Oracle Endeca Workbench and Experience Manager.
- **User profiles** - User profiles have been replaced by User segments. For details, see *Oracle Endeca Workbench* on page 58.

Theses features are ignored in Developer Studio and will be removed in a later release.

Oracle Endeca Workbench

Versioning

Oracle Endeca Workbench has the same version number as Tools and Frameworks. For example, in this release, both are version 3.1.0.

Installation changes

In the Tools and Frameworks 3.1.0 release, Oracle Endeca Workbench has been repackaged as part of Tools and Frameworks. Oracle Endeca Workbench is no longer a separate installation.

Silent installation changes

Oracle Endeca Workbench does not support silent installation as it did in 2.1.x. As described earlier, Oracle Endeca Workbench has been repackaged as part of Tools and Frameworks 3.1.0.

Tools and Frameworks does not have an installation program, and as a consequence, Oracle Endeca Workbench does not support silent installation. However, if a 2.1.x implementation installed Workbench silently, you can modify the installation scripts to perform a similar set of steps in 3.1.0 by doing the following:

1. Unzip the Tools and Frameworks installation package.
2. Install the Endeca Tools Service by running the `ToolsAndFrameworks\<version>\server\bin\install_service` script.
3. Start the Endeca Tools Service by running the `ToolsAndFrameworks\<version>\server\bin\start_service` script.

Rule Manager and Experience Manager are mutually exclusive

In Workbench 2.1.x, Rule Manager was always installed as an extension to Workbench. Optionally, you could also install Page Builder 2.1.x and have both running as extensions to Workbench. In the Tools and Frameworks 3.1.0 release, Oracle Endeca Workbench can have either Rule Manager or Experience Manager but not both installed.

Resource locking

In previous releases, Workbench locked resources at the page level and rule group level for each user. In other words, a user logged in to Workbench and locked one rule group or page at a time, edited that group or page, and then selected a different rule group or page. At that point, Workbench released the resource lock so other users could access the rule group or page. This resource locking model caused access problems for multiple users of Workbench.

In the Tools and Frameworks 3.1.0 release, Oracle Endeca Workbench employs a much more fine-grained locking model that allows multiple users to log in and access the same rule group or the same page. Workbench warns users if they happen to edit the same rule and then displays options to reconcile the conflict.

Pages removed from Workbench

The following table lists the pages that have been removed from Workbench in the Tools and Frameworks 3.1.0 release.

Page Name	Description of change
Dimension Order	Dimension groups and the **Dimension Order** page have been removed. In 3.1.0 and later, you can modify dimension order in an application by reordering cartridges on a page in Experience Manager. You can further reorder dimensions by sorting them within a cartridge.
Stop Words	The **Stop Words** page has been removed. However, stop words are still supported in an Endeca application. As a work around,

Page Name	Description of change
	you can modify stop words using the Stop Words editor in Developer Studio.
Phrases	The **Phrases** page has been removed. However, automatic phrasing is still supported in an Endeca application. As a work around, you can modify phrases using the Automatic Phrases editor in Developer Studio.

Workbench page menu has moved

In Oracle Endeca Workbench 2.1.x, there was a menu of pages available in the left margin of Workbench. In the Tools and Frameworks 3.1.0 release, the menu of pages is now a drop-down menu that displays when you click the down arrows in the header of the page.

User Profiles have been renamed to User Segments

In Oracle Endeca Workbench 2.1.x, you could create dynamic business rules that can include a user-profile trigger.

In the Tools and Frameworks 3.1.0 release, User Profiles have been renamed to User Segments to better describe that pages and rules can be triggered for just specific user segments. You create them on the **User Segments** page of Workbench and add them to a content item using the **Select User Segments** dialog of Experience Manager.

Note: User Profiles have not been removed from Endeca Developer Studio. Workbench and Experience Manager ignore any User Profiles you configure in Developer Studio.

The `ENDECA_TOOLS_CONF` value has changed

The `ENDECA_TOOLS_CONF` environment variable sets the path to the `Workbench\workspace` directory. The value of `ENDECA_TOOLS_CONF` has changed in Tools and Frameworks 3.1.0 as shown in this table:

Version	Default Value
Workbench 2.1.x	`C:\Endeca\Workbench\workspace` (on Windows) `endeca/Workbench/workspace` (on UNIX)
Tools and Frameworks 3.1.0	`C:\Endeca\ToolsAndFrameworks\3.1.0\server\workspace` (on Windows) `endeca/ToolsAndFrameworks/3.1.0/server/workspace` (on UNIX)

Downloading the instance configuration

The option to download the instance configuration files as a single archive (`instconfig.zip`) has been removed. to perform a similar back up operation, you can run the `export_site` script in the `<app dir>\control` directory.

Adding and removing an application

The ability to add and remove an application has been removed from the **EAC Admin** page in Oracle Endeca Workbench 3.1.0. In 3.1.0 and later, you provision an application using the Deployment Template's `initialize_services` script.

Documentation removal

The Oracle Endeca Workbench documentation set has been merged into the Tools and Frameworks documentation set. In the 3.1.0 release, this includes only the *Workbench Administrator's Guide*. The *Oracle Endeca Workbench Help* and the *Oracle Endeca Workbench User's Guide* are not available

Endeca Assembler

Endeca Query Language

In the Tools and Frameworks 3.1.0 release, the Endeca Assembler does not support the Endeca Query Language.

Disabled refinements

In the Tools and Frameworks 3.1.0 release, the Endeca Assembler does not support disabled refinements.

Changes to Assembler cartridges

Here are the cartridges available in 3.1.0:

Search Box cartridge

Auto-suggest search results cartridge

Dimension Search cartridge

Search Adjustments cartridge

Keyword redirects cartridge

Refinement Menu cartridge

Navigation Container

Breadcrumbs cartridge

Results List cartridge

Record details cartridge

Record Spotlight cartridge

Media Banner cartridge

Page Builder and Experience Manager

In the Tools and Frameworks 3.1.0 release, Page Builder has been renamed to Experience Manager to better reflect that business users are managing dynamic online experiences across all channels and can control more granular sections of the application user's experience.

Similarly, the Page Builder SDK has been renamed to the Experience Manager Editor SDK.

Installation changes

In the Tools and Frameworks 3.1.0 release, the Experience Manager Editor SDK has been repackaged as part of Tools and Frameworks. It is no longer a separate installation.

Versioning

The Experience Manager Editor SDK has the same version number as Tools and Frameworks. For example, in this release, both are version 3.1.0.

Documentation changes

The *Experience Manager Editor Developer's Guide* has been merged into the Tools and Frameworks documentation set.

The Deployment Template Module for Product Catalog Integration

Installation changes

In the Tools and Frameworks 3.1.0 release, the Deployment Template Module for Product Catalog Integration has been repackaged as part of Tools and Frameworks. The Deployment Template Module for Product Catalog Integration is no longer a separate installation.

By default, the Deployment Template Module for Product Catalog Integration is installed into `C:\Endeca\ToolsAndFrameworks\<version>\reference\discover-data-pci` (on Windows) and `usr/local/endeca/ToolsAndFrameworks/<version>/reference/discover-data-pci` (on UNIX).

Forge Configuration Generation Adapter defaults

The Forge Configuration Generation Adapter uses a `matchMode` property to specify the default matching behavior when creating new dimensions. In the previous release, the `matchMode` property was set to `AUTO_GEN`. In this release, the default value is set to `NORMAL`.

For details about how to change property values, see "Globally Modifying the Default Values for Schema Attributes" in the *Deployment Template Module for Product Catalog Integration - Usage Guide*.

Schema.csv defaults

In the `schema.csv` file, the default value of `attribute.dimension.autogen` has been changed from `true` to `false`. (This is an optional Boolean value that indicates whether the dimension has been autogenerated by Forge.)

For details about schema values, see "Format of the Schema CSV File" in the *Deployment Template Module for Product Catalog Integration - Usage Guide*.

Documentation changes

The *Deployment Template Module for Product Catalog Integration - Usage Guide* is distributed as part of the Tools and Frameworks documentation set.

The URL Optimization API

In the Tools and Frameworks 3.1.0 release, the URL Optimization API has been repackaged as part the Endeca Assembler API, and it ships as part of Tools and Frameworks. The URL Optimization API is no longer a separate installation.

Documentation changes

The *Oracle Endeca Commerce URL Optimization API for Java Developer's Guide* is distributed as part of the Tools and Frameworks documentation set. The *URL Optimization API Reference (Javadoc)* is installed into the *Endeca Assembler API Reference (Javadoc)*.

The emgr_update utility is no longer public

In previous releases, the `emgr_update` utility assisted you in updating the instance configuration of an Endeca implementation system. In the Tools and Frameworks 3.1.0 release, the `emgr_update` utility is no longer the public interface to perform such administrative system operations.

Most administrative operations are performed using Deployment Template scripts that are stored in the `<app name>\control` directory of an application.

Note: The `emgr_update` utility is still installed and used internally by Endeca components.

The Shuffle setting

The **Shuffle** setting for dynamic business rules is no longer supported in Assembler applications. This setting is available in Developer Studio. (If checked, the option shuffles the promoted records for the associated rule.) The setting is ignored in rules that have the **Shuffle** option selected.

Index