Oracle Endeca Commerce

Tools and Frameworks Migration Guide

Version 11.0 • January 2014

ORACLE®

COMMERCE

Contents

Copyright and disclaimer

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Preface

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 into results that can be rendered for display.

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 determine the conditions for displaying content in response to any search, category selection, or facet refinement.

About this guide

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

Complete the steps described for your migration arc, as well as any steps in the Required Changes section that apply to features you use. Changes described in Behavioral Changes either do not require action on your part, or are optional.

Who should use this guide

This guide is intended for application developers and administrators 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 on 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 Supported Environments and Compatibility* available on the Oracle Technology Network.

Prerequisite upgrades

Before upgrading Tools and Frameworks, you must first upgrade the MDEX Engine and Platform Services to compatible versions. For information on package compatibilities, see *Oracle Endeca Commerce Supported Environments and Compatibility*. For information about how to migrate the MDEX Engine and Platform Services, refer to the *MDEX Engine Migration Guide* and the *Platform Services Migration Guide*.

Upgrade paths

There is one supported upgrade path:

- From Oracle Endeca Tools and Frameworks 3.1.2 to Oracle Endeca Tools and Frameworks 11.0

Unsupported upgrade paths

If you are migrating to Oracle Endeca Tools and Frameworks 11.0 from a version other than those listed above, you must first migrate to the previous release (3.1.2).

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 Tools and Frameworks from 3.1.2 to 11.0

The following section lists the steps for migrating from Tools and Frameworks 3.1.2 to Tools and Frameworks 11.0.

About migrating from 3.1.2 to 11.0

To migrate your Tools and Frameworks installation from 3.1.2 to 11.0, you first create backups of your configuration files and application content, then upgrade to the latest Tools and Frameworks package. Afterwards, you can restore Workbench configuration from your backups, and update application configuration from a combination of backups and migration scripts.

Migrating from Tools and Frameworks 3.1.2 to 11.0 consists of the steps below. They are described in detail in the following sections:

1. Create backups:
    a. Back up Workbench configuration from the `%ENDECA_TOOLS_CONF%\conf` directory.
    b. Back up application state from the Workbench `%ENDECA_TOOLS_CONF%\state` directory.
    c. Back up the Endeca Configuration Repository by running the `export-site` script for each application you are migrating.
    d. Back up any Forge data ingest configuration by creating a copy of the `config\pipeline` directory in your application.
    e. Optionally, back up each deployed application directory, for example, `C:\Endeca\apps\Discover`. You will need the files on disk to migrate application content to Tools and Frameworks 11.0.
    f. If you are migrating a joint ATG and Endeca application, create a list of ATG user segments prior to migrating your ATG components. You can retrieve this list by navigating to `http://<ATG host>:<ATG port>/dyn/admin/nucleus/atg/registry/RepositoryGroups`.
    g. Stop your Tools and Frameworks 3.1.2 Workbench, but do not uninstall it. You will need the export scripts included with Tools and Frameworks 11.0 to export user information from version 3.1.2.
2. Upgrade Tools and Frameworks to version 11.0:
    a. Install Tools and Frameworks 11.0 as documented in the *Tools and Frameworks Installation Guide*.
    b. Restart your 3.1.2 Workbench installation and export your Workbench user information.
    c. Uninstall Tools and Frameworks 3.1.2 as documented in the *Tools and Frameworks Installation Guide*.
3. Restore Workbench configuration:
    a. Restore Workbench configuration from the backups created in Step 1a.

b. Restore the application state directories in Workbench from the backups created in Step 1b.

4. Restore each of your applications:
   a. Configure the the `migrate-workbench` script. Optionally, you may also configure logging for the migration script.
   b. Migrate application configuration by running the `migrate-workbench` script.

      The script converts Thesaurus entries, cartridge templates, and application content. The new configuration is stored as a modified version of the ECR export file and an accompanying set of cartridge template files.

   c. Restore Forge configuration from backups.
   d. Deploy and initialize the 11.0 version of your application.
   e. Restore the Endeca Configuration Repository content from the modified export file created in Step 4b by running the `import-site` script.
   f. Restore Workbench user information using the `import_users` script.
   g. Upload the migrated cartridge templates by moving them to the `cartridge_templates` directory in your application and running the `set_templates` script.
   h. Upload editor configuration by running the `set_editors_config` script.
   i. Run a baseline update to load data into the Dgraph and start it.
   j. Update the Oracle Commerce JAR files in your application to version 11.0.
   k. Verify that Workbench configuration and application content are present in Workbench. Optionally, promote content to your live servers.
   l. Repeat the substeps above for each application you are migrating.
5. Import your Workbench users from the information you exported in Step 2b.

Creating backups

Before upgrading your Tools and Frameworks installation, you must back up your Workbench and application configuration and content.

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

File name	Description
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

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:

   ```
   export_site.bat C:\migration\Discover\ECR-backups\discover-21-12-2013.xml
   ```

   On UNIX:

   ```
   ./export_site.sh ../migration/Discover/ECR-backups/discover-21-12-2013.xml
   ```

If no file name is provided, it defaults to a file named according to the pattern `<sitename>-yyyy-mm-dd_time.xml` in the working directory.

Backing up Forge pipeline configuration

Forge configuration is stored in the `<app dir>\config\pipeline` directory of your deployed application on disk.

To back up Forge pipeline configuration, copy the `<app dir>\config\pipeline` directory and its contents to another location. This directory contains data ingest pipeline configuration for search interfaces, navigable dimensions, and other configuration authored with Developer Studio.

Backing up the application directory

Before installing Tools and Frameworks 11.0, you should back up your application directory, for example: `C:\Endeca\apps\Discover`.

While these files are not removed when uninstalling a previous version of Tools and Frameworks and upgrading to version 11.0, you should have a backup available since they are used as migration source files for the migration scripts.

To back up the application directory:

1. Locate the application in your Endeca application directory.
   For example, `C:\Endeca\Apps\Discover`.
2. Copy the application folder and its contents to a temporary directory that is outside the Endeca installation directory.
   For example, you might copy it to `C:\migration\Discover`.
3. Open a command prompt window and remove the application from the EAC Central Server by running the `eaccmd` utility and the `remove-app` command.
   For example, `eaccmd remove-app --force --app Discover`
4. Delete the application directory from your Endeca application directory.

Backing up ATG user segments to a text file

If you are migrating a joint ATG and Endeca application, you should create a list of your ATG user segments prior to uninstalling the previous versions of your ATG components.

These steps are only necessary if you are migrating an integrated ATG and Endeca implementation.

To back up ATG user segments:

1. Retrieve the list of user segments from the Nucleus repository by navigating to `http://<ATG host>:<ATG port>/dyn/admin/nucleus/atg/registry/RepositoryGroups`.
2. Copy the list of user segments under **Directory Listing** > **UserProfiles**.
3. Paste the list into a text file, with each user segment on its own line.
4. Save the text file to a location where the Tools and Frameworks migration utility can access it in later steps.
   For example, `C:\migration\CRS\atgSegments.txt`.

Installing Tools and Frameworks 11.0

After backing up Workbench and application configuration and content, stop your Tools and Frameworks 3.1.2 installation and install Tools and Frameworks 11.0. Do not install the 11.0 version of the Endeca Tools Service or uninstall version 3.1.2.

Before installing Tools and Frameworks 11.0, you must back up your existing Workbench and application configuration, as well as your application directory.

User migration from 3.1.2 to 11.0 requires a running instance of Tools and Frameworks 3.1.2; for this reason, do not uninstall the 3.1.2 package or install the 11.0 version of the Endeca Tools Service (on Windows).

To install Tools and Frameworks 11.0:

1. Stop your 3.1.2 Tools and Frameworks installation:

Option	Description
On Windows:	Stop the Endeca Tools Service or run the `ToolsAndFrameworks\3.1.2\server\bin\stop.bat` file.
On UNIX:	Run the `ToolsAndFrameworks/3.1.2/server/bin/workbench-init-d.sh` shell script with the `/sbin/service workbench stop` command.

2. Install Tools and Frameworks 11.0. See the *Tools and Frameworks Installation Guide* and perform the tasks in "Installing Oracle Endeca Tools and Frameworks on Windows" or "Installing Oracle Endeca Tools and Frameworks on UNIX".

Important: Do not set up the Endeca Tools Service on Windows or the `workbench` service on UNIX. You must run the 3.1.2 version of Workbench to export user information.

Once you have installed Tools and Frameworks 11.0, you can export your user information from Tools and Frameworks 3.1.2.

Exporting Workbench users from Tools and Frameworks 3.1.2

You can use the `export_users` script in Tools and Frameworks 11.0 to export user information from your 3.1.2 Workbench installation.

You must have Tools and Frameworks 11.0 installed on your system as well as your old Tools and Frameworks 3.1.2 installation in order to export users for migration.

To export Workbench users from Tools and Frameworks 3.1.2:

1. Start your 3.1.2 Tools and Frameworks installation:

Option	Description
On Windows:	Start the Endeca Tools Service or run the `ToolsAndFrameworks\3.1.2\server\bin\run.bat` file.
On UNIX:	Run the `ToolsAndFrameworks/3.1.2/server/bin/workbench-init-d.sh` shell script with the `/sbin/service workbench start` command.

2. Configure the `export_users` script:

a) Navigate to the `ToolsAndFrameworks\11.0.0\admin\conf` directory.
b) Open the `export_users.properties` file in a text editor.
c) Set the following properties:

Property	Description
`source.version`	The version of Tools and Frameworks you are migrating from.
`source.workbench.host`	The host machine of the Workbench instance you are exporting from.
`source.workbench.port`	The port of the Workbench instance you are exporting from.
`destination.directory`	The directory of the Workbench instance you are migrating to.

For example:

```
source.version = 3.1.2
source.workbench.host=myHost.myDomain.com
source.workbench.port=8006
destination.directory=C:/Endeca/ToolsAndFrameworks/11.0
```

d) Save and close the file.

3. Navigate to the `ToolsAndFrameworks\11.0.0\admin\bin` directory.
4. Run the `export_users` script with the following parameters:
   - `--config` — Required. The path to the `export_users.properties` file.
   - `--output` — Optional. The path of the output file. If you do not specify this option, the output file is `.\user_<timestamp>.json`.

   For example:

   ```
   > export_users.bat --config ..\conf\export_users.properties
   ```

   This installs a new bundle to the 3.1.2 Workbench and exports users in a format readable by the 11.0 Workbench.
5. Confirm that the JSON file exists and contains the expected users.

   Note the following information as you review the results:
   - All administrator users and groups have an *admin* property set to `true` and do not have permission attributes.
   - LDAP users and groups have a *principalSource* value of `LDAP`. All others have the default value of `Workbench`.

   Sample user :

```
{
    "id": "mmartin",
    "firstName": "melanie",
    "lastName": "martin",
    "email": "mmartin@example.com",
    "principalSource": "WORKBENCH",
    "admin": false,
    "permissions": [
        {
            "application": "Discover",
                "tools": [
                    "extension1"
                ],
```

```
        }
    ]
}
```

Sample group:

```
{
    "id": "global",
    "groupName": "global merchandising",
    "email": "global@example.com",
    "principalSource": "LDAP",
    "permissions": [
        {
            "application": "Discover",
            "tools": [],
        }
    ]
}
```

You can check the script log at `Endeca\ToolsAndFrameworks\11.0.0\admin\logs\export_users.log`.

After you have exported your Workbench users, uninstall Tools and Frameworks 3.1.2. See version 3.1.2 of the *Tools and Frameworks Installation Guide* and perform the tasks in "Uninstalling Oracle Endeca Tools and Frameworks".

Restoring Workbench configuration

After updating to Tools and Frameworks 11.0, you can restore your Workbench configuration from backups. You must also restore the application state directories.

If you plan to use the Endeca Tools Service on Windows or the `workbench` service on UNIX to manage the Workbench application container, follow the steps in the *Tools and Frameworks Installation Guide* to install it before proceeding further.

Restoring a backup of the Workbench configuration files

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

The steps below assume the Endeca Tools Service (on Windows) or the `workbench` service (on UNIX) is configured and running.

After updating to Tools and Frameworks 11.0, you can restore your Workbench configuration from backups. This includes Workbench extensions, menu configuration, and authentication settings.

Note: User profiles must be re-created manually.

To restore a backup of the Workbench configuration files:

1. Stop the Endeca Tools Service.
2. Open your configuration backup files.
   These are the files created from *Backing up the Workbench configuration files* on page 12.

3. Manually merge any configuration changes into the files located at `%ENDECA_TOOLS_CONF%\conf` (on Windows) or `$ENDECA_TOOLS_CONF/conf` (on UNIX).
4. Save and close the files.
5. Start the Endeca Tools Service.

Restoring a backup of the application state

Restore backups of the application state directories, `webstudiostore` and `emanager`, by copying them into the new installation of Workbench.

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

Restoring Forge pipeline configuration

You can restore Forge pipeline configuration by copying the backup of the `config\pipeline` directory over the `<app dir>\config\pipeline` directory of your migrated application.

Migrating Endeca applications from 3.1.2 to 11.0

To migrate an Endeca application from Tools and Frameworks 3.1.2 to 11.0, you must restore the application configuration from backups and run migration scripts to convert templates and content for 11.0. Repeat these steps for each application you are migrating.

Configuring the migrate-workbench script

Before running the `migrate-workbench` script, you must modify the `config.properties` file with source and destination directories for migration.

To configure the `migrate-workbench` script:

1. Navigate to `ToolsAndFrameworks\<version>\migration\workbench\3.1.2-11.0\conf`.
2. Open `config.properties` in a text editor.
3. Set `endeca.source.appDir` to the location of your 3.1.2 application on disk.
   For example, `C:/Endeca/apps/Discover`.
4. Optionally, set `endeca.source.templateDir` to the location of your 3.1.2 cartridge templates on disk. If you leave this blank, the value is determined relative to your entry for `endeca.source.appDir`.

These files should be present in your backed up application directory. For example, in the default Discover Electronics reference application, these files are located in `C:\Endeca\apps\Discover\config\cartridge_templates`.

Important: Paths in the config.properties file must use a forward slash (`/`) or escaped backslashes (`\\`) as file separators. A single backslash character will not work correctly.

If you instead created a backup of the directory, use the path to the backup.

5. Set `endeca.target.templateDir` to an empty output directory where you wish to store the migrated templates.

   If this directory does not exist, it is created when you run the script.
6. Set `endeca.source.exportedSiteFile` to the exported site file previously created from running the `export_site` script on your 3.1.2 application.
   For example, `C:/migration/Discover/ECR-backups/discover-21-12-2013.xml`.
7. Set `endeca.target.exportedSiteFile` the output file where you wish to store the migrated site content.
   For example, `C:/migration/Discover/11.0-export.xml`.
8. If you are migrating a joint ATG and Endeca application, set `endeca.atgSegmentsFile` to the plain text file where you saved the list of ATG user segments.
   For example, `C:/migration/CRS/atgSegments.txt`.

   

   Note: To differentiate ATG user segments from Endeca user segments, ATG segments are prefixed with `UserState.profileGroup:`. Because ATG user segments are retrieved directly from the ATG server in 11.0, the migration process removes any Endeca segments with the same name as an ATG segment.
9. Save and close the file.

After modifying the `config.properties` file, you can optionally configure logging for the script.

Configuring logging for the migrate-workbench script

You can configure the log level and output directory for the `migrate-workbench` logs by modifying the `logging.properties` file.

The `migrate-workbench` script logs its migration operations. By default, logs are stored in a `migration.log` file within the directory from which you execute the migration script.

To configure logging for the `migrate-workbench` script:

1. Navigate to `ToolsAndFrameworks\<version>\migration\workbench\3.1.2-11.0\conf`.
2. Open `logging.properties` in a text editor.
3. Set `java.util.logging.FileHandler.pattern` to the desired logging output directory.
4. Set `java.util.logging.FileHandler.level` to the desired log level.

   The script provides information at the following logging levels:

   - `SEVERE` — Logs errors that terminate the migration process.
   - `WARNING` — Logs information about problems during migration that are not severe enough to stop the process, but may require action from the user to repair.
   - `INFO` — Logs the status of the migration process.

- `FINER` — Logs additional information about the migration process, such as configuration information and template metadata.
- `FINEST` — Logs verbose information about the migration process, such as source and target content XML and template files.

For additional information, see the Oracle documentation for the `java.util.logging.Level` class.

5. Save and close the file.

Running the migrate-workbench script

Once the `migrate-workbench` script is configured, you can run it to convert application configuration to the 11.0 format.

Before performing this task, ensure you have backed up your 3.1.2 application directory and have it available on the local file system where you are running the `migrate-workbench` script.

The `migrate-workbench` script converts the following features to 11.0 format:

- Thesaurus entries
- Application pages
- Content items
- Content collections

Note: Content collections are converted to content folders as part of migration. Because content folders do not store a rule limit restriction, before conversion the rule limits on any dynamic slots are set to either their current rule limit, or the limit of the collection they refer to, whichever is lower.

- Cartridge templates

Note: Migrated cartridge templates are created in deprecated 11.0 format. Refer to the *Assembler Application Developer's Guide* for the latest cartridge template format.

- Automatic phrases

  There are no phrase entries in Tools and Frameworks 3.1.2. The MDEX phrases pipeline file is used in the migration, instead. Phrases in the pipeline file are added to the migrated site, but any dimension imports found in the phrases pipeline file are not included because that feature is no longer supported.

It also copies 3.1.2 features that do not require conversion, such as keyword redirects, user segments, and any existing Preview settings.

To migrate Workbench configuration:

1. Open a command prompt window and navigate to `Endeca\ToolsAndFrameworks\<version>\migration\workbench\3.1.2-11.0\bin` .
2. Run the `migrate-workbench` batch or shell script.

   The script logs a message to the console indicating succesful migration and the location of the output files:

```
C:\Endeca\ToolsAndFrameworks\11.0.0\migration\workbench\3.1.2-11.0\bin>
migrate-workbench.bat
Jul 23, 2013 3:23:00 PM com.endeca.migration.MigrationLauncher logSuccessMessage
INFO: Migration completed.
Jul 23, 2013 3:23:00 PM com.endeca.migration.MigrationLauncher logSuccessMessage
INFO: Migrated cartridge templates are at path:
```

```
C:/migration/Discover/11.0-cartridge_templates
Migrated export file is at path: C:/migration/Discover/11.0-export.xml
```

3. Navigate to the output paths and confirm that a cartridge template files and an export XML file are present.

Configuring communication with the ATG user segments server

If you are running an integrated ATG and Endeca application that does not use the Commerce Reference Store Deployment Template descriptor file (via CIM), you must modify your ATG services definition in the EAC application to specify the URL of your ATG server.

Note: The following steps are not necessary if you are not running an integrated application, or if you are using the CRS Deployment Template descriptor.

To configure communication with the ATG server:

1. Navigate to the `config\editors_config` directory of your deployed application on disk.
2. Open `atgServices.json.`
3. Set the `profileGroupsConnectionUrl` property to the JSON response for the `getAllProfileGroups` operation on the ATG publishing server.
   For example:

```
{
  profileGroupsConnectionUrl:"http://<ATG host>:<ATG port>/rest/model/atg/user¬
profiling/ProfileGroupsActor/getAllProfileGroups?atg-rest-output=json"
}
```

Note: Oracle recommends specifying the host and port for the ATG Content Administration server. This ensures that Workbench retrieves only the verified, published user segments for an application.

4. Save and close the file.

The `initialize_services` script publishes the updated file when you run it to initialize the application with the EAC. To manually publish the updated file at a later time, run the `set_editors_config` script.

Deploying and initializing the 11.0 application

After running the `migrate-workbench` script, use the Deployment Template to deploy an 11.0 version of your application, then run the `initialize_services` script to provision it in the Endeca Application Controller.

To deploy an empty destination application:

1. Open the command prompt.
2. Navigate to `ToolsAndFrameworks\<version>\deployment_template\bin`.
3. From the `bin` directory, run the `deploy` script with the `--app` flag and an argument that specifies the path to the `deploy.xml` deployment descriptor file.
   For example:

```
C:\Endeca\ToolsAndFrameworks\11.0.0\deployment_template\bin>deploy
--app C:\Endeca\ToolsAndFrameworks\11.0.0\reference\discover-data\deploy.xml
```

4. Unless your environment requires you to use different ports, accept the default values during the deployment process. You must specify the same application name that you used in your 3.1.2 application.

For details on running the `deploy` script, see the *Tools and Frameworks Deployment Template Usage Guide*.

5. Navigate to the `control` directory of the deployed application.
   For example: `C:\Endeca\Apps\Discover\control`.
6. Run the `initialize_services` script.
   This provisions the application within the Endeca Application Controller and creates the necessary structure in the Endeca Configuration Repository.

Deploying and provisioning the application enables you to upload the modified export file to the ECR. See the following section for details.

Uploading migrated application configuration and templates to the Endeca Configuration Repository

Running the `import_site` script uploads your migrated application content to the Endeca Configuration repository. The `set_templates` script uploads the migrated cartridge templates.

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

To upload migrated 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 modified export file produced by the `migrate-workbench` script..
   For example:
   ```
   C:\Endeca\apps\Discover\control>import_site.bat
   C:\migration\Discover\11.0.0-export.xml
   ```
   The following prompt appears:
   ```
   "Application '<app name>' already exists in IFCR.
   Delete existing content and continue? [Y/N]:"
   ```
3. Enter `Y` to proceed.
4. Confirm that the import operation completes succesfully.
5. Navigate to the `cartridge_templates` directory in your deployed application on disk.
   For example, `C:\Endeca\apps\Discover\config\cartridge_templates`.
6. Delete any existing cartridge templates.
7. Navigate to the cartridge template output directory for the `migrate-workbench` script.
   For example, `C:\migration\Discover\11.0.0-cartridge_templates`.
8. Copy the migrated templates to the `cartridge_templates` directory in your deployed application.
9. Navigate to the `control` directory of your deployed application.
10. Run the `set_templates` script.
    This uploads the migrated templates to your application.
11. Run the `set_editors_config` script.
    This uploads editor configuration to your application.

After updating your ECR content and templates, you must run a baseline update to index data and start the Dgraph for your newly-deployed application.

Running a baseline update

After you have deployed your application and uploaded the migrated content and configuration, you must initialize and start the Dgraph to make MDEX Engine data available.

To run a baseline update:

1. Navigate to the `control` directory of your deployed application.
   For example, `C:\Endeca\apps\Discover\control`.
2. Run the `load_baseline_test_data` script.
3. Run the `baseline_update` script.

Updating Endeca application JAR files to version 11.0

After restoring your content, you must update your application with 11.0 JAR files.

To update Endeca application JAR files:

1. Navigate to the `%ENDECA_TOOLS_ROOT%\11.0.0\deployment_template\app-templates\common\config\lib\java` folder (on Windows) or `$ENDECA_TOOLS_ROOT/11.0/deployment_template/app-templates/common/config/lib/java` (on UNIX).
2. Copy the 11.0 JAR files:
   - `eacComponents-11.0.jar`
   - `eacToolkit-11.0.jar`
3. Navigate to the library folder of your Web application.
   This may be the `WEB-INF\lib` directory, or another location where you have chosen to store JAR files.
4. Paste the JAR files that you copied into the folder.
5. Delete the previous versions of the JAR files:
   - `eacComponents-3.1.2.jar`
   - `eacToolkit-3.1.2.jar`

Importing 3.1.2 Workbench users to Tools and Frameworks 11.0

You can use the `import_users` script to import user information from a JSON format to your Workbench installation.

To import 3.1.2 Workbench users to Tools and Frameworks 11.0:

1. Configure the `import_users` script:
   a) Navigate to the `ToolsAndFrameworks\11.0.0\admin\conf` directory.
   b) Open the `import_users.properties` file in a text editor.

c) Set the `dest.workbench.host` and `dest.workbench.port` properties to the host and port of your Tools and Frameworks 11.0 Workbench.

By default, these values are set to `localhost` and `8006`, respectively.

For example:

```
dest.workbench.host=myhost.mycompany.com
dest.workbench.port=8006
```

d) Set the `abort.on.duplicate.users` Boolean property:

- `true` — If duplicate users are detected, the script stops and creates a `duplicate_users.log` log file that lists all duplicate user names.
- `false` — If duplicate users are detected, the script attempts to merge them into a single entry.

For example:

```
abort.on.duplicate.users=false
```

e) Comment out the `tools.mapping.file` property:

```
#-------- location of tools mapping (required if source.version 2.1.x & 3.1.0)
#tools.mapping.file=<path-to-tools-mapping-json>
```

f) Optionally, set the `convert.ldap` property to `true` if you wish to convert all users to LDAP users during import.
For example:

```
convert.ldap=true
```

g) Save and close the file.

2. Navigate to the `ToolsAndFrameworks\11.0.0\admin\bin` directory.
3. Run the `import_users` script with the following parameters:

- `--admin-user` — Required. The administrator user name in Workbench.
- `--admin-password` — Required. The administrator password in Workbench.
- `--input` - Required. The path to the user data file.
- `--config` - Requred. The path to the `import_users.properties` file.
- `--default-user-password` — The password value for any users that do not have a password set in the exported users file.
- `--single-app` — Optional. A single application for which to import users.

For example:

```
> import_users.bat --admin-user admin --admin-password kinginthecastle
--input user_20131201.json --config ..\conf\import_users.properties
--default-user-password CHANGEME --single-app Discover
```

If a name matches an existing name but with characters in a different case (JOHN_Doe and John_Doe) and the `abort.on.duplicate.users` property is set to `true`, the script stops. Invalid entries are logged to `ToolsAndFrameworks\<vesrion>\admin\logs\import_validation_failed.log`. The main log file is output to `ToolsAndFrameworks\<version>\admin\logs\import_users.log`.

Verifying a migrated application

Once you have finished migrating Workbench and application configuration, log in to Workbench to confirm that everything is present and functioning as expected.

To verify that your application migrated correctly:

1. Navigate to the URL for your application and confirm that it is running.
   For example, the authoring version of the reference application is typically available from `http://localhost:8006/discover-authoring`.
2. Log in to Workbench and verify that the following are correctly configured:
   - Thesaurus entries
   - Keyword redirects
   - User segments
3. Open Experience Manager and confirm that the following are present:
   - Cartridge templates
   - Application pages
   - Application content (content items)
4. If your application has been successfully migrated, you may optionally promote the content and configuration to your live servers.
   In the Discover Electronics reference application, this is accomplished by running the `promote_content` script in the `control` directory.

If you encounter any issues with migration, see *Troubleshooting migration errors* on page 25.

Troubleshooting migration errors

The `migrate-workbench` script outputs error messages on failure. Refer to the table below if you are encountering issues running the script.

Error	Action
Could not find `config.properties`	Make sure that a `config.properties` file exists in the `migration\workbench\3.1.2-11.0\conf` directory of your Tools and Frameworks installation.
Could not find `logging.properties`	Make sure that a `logging.properties` file exists in the `migration\workbench\3.1.2-11.0\conf` directory of your Tools and Frameworks installation.
Source cartridge template directory path cannot be empty.	Make sure that you have entered a value for the `endeca.source.templateDir` property in `config.properties`.
Source cartridge template must point to a valid directory.	Make sure that the `endeca.source.templateDir` property is set to a non-empty 3.1.2 cartridge template directory.
Target cartridge template directory path cannot be empty.	Make sure that you have entered a value for the `endeca.target.templateDir` property in `config.properties`.

Error	Action
Target cartridge template directory must be empty.	Make sure that the `endeca.target.templateDir` property is set to an empty directory.
Source and target cartridge template directory paths cannot be same.	Make sure that `endeca.source.templateDir` and `endeca.target.templateDir` are not set to the same path.
Source export file path cannot be empty.	Make sure that you have entered a value for the `endeca.source.exportedSiteFile` property in `config.properties`.
Source export file must exist.	Make sure that the `endeca.source.exportedSiteFile` property is set to the exported 3.1.2 XML file.
Target export file path cannot be empty.	Make sure that you have entered a value for the `endeca.target.exportedSiteFile` property in `config.properties`.
Target export file already exists.	Make sure that the `endeca.target.exportedSiteFile` property is set to a file path that does not already exist.
Source and target export file paths cannot be same.	Make sure that `endeca.source.exportedSiteFile` and `endeca.target.exportedSiteFile` are not set to the same path.
Error while migrating cartridge template: `<templateFilePath>`. Could not find templateType for templateId: `<templateId>`	Make sure that cartridge-template at `<templateFilePath>` is valid.
Error while parsing content.xml for `<content>`. Could not find templateType for templateId: `<templateId>`	Make sure that the `endeca.source.templateDir` directory in `config.properties` contains a valid cartridge template for each `<templateId>` referenced in your content items.
Error while parsing `appConfig` file.	Make sure that the `appConfig.xml` file in `{appDir}/config/` script is correct XML
Cannot find `appConfig` file.	Make sure that the `appConfig.xml` file is in `{appDir}/config/script` directory.
Cannot read appConfig file.	Make sure that the migrator has permission to read the appConfig file.
Cannot find phrases pipeline file.	Make sure that the file `{AppName}.phrases.xml` exists in `{appDir}/config` directory.
A phrases node already exists.	Make sure that you have not performed migration twice, because the phrases node should not exist in the 3.1.2 exported site.

Chapter 3

Required Changes

The following section lists required changes as of version 11.0. You must make the changes specified in this section if the changes apply to your application.

Deprecated features

The features listed below are deprecated in the Tools and Frameworks 11.0 release. They are still fully supported.

Rule Manager

The Rule Manager tool in Workbench is deprecated in the Tools and Frameworks 11.0 release. Oracle recommends adopting Tools and Frameworks with Experience Manager for new projects.

Experience Manager Editor SDK

The Experience Manager Editor SDK is deprecated in the Tools and Frameworks 11.0 release. Consult your Professional Services representative for guidance if you wish to develop custom editors for your applications.

Assembler API Changes

For details on Assembler API changes that deprecate classes or properties, see the following topics:

- *Workbench content source changes* on page 28
- *Performance logging changes* on page 28
- *The Dynamic Slot cartridge* on page 28

Deployment Template Module for Product Catalog Integration

The Deployment Template Module for Product Catalog Integration is deprecated in the Tools and Frameworks 11.0 release. Oracle recommends using the Discover CAS-based deployment template described in the *CAS Quick Start Guide* and the *CAS Developer's Guide*.

Workbench content source changes

Assembler API changes

The following properties on the `WorkbenchContentSource` class have been deprecated and their functionality has been incorporated into the `EcrStoreFactory` class:

- `isAuthoring`
- `host`
- `clientPort`
- `serverPort`

There is also a new `storefront` property that defines the factory type, either *ecrStoreFactory* or *fileStoreFactory*. For example, in the Discover Electronics authoring application the content source has the following properties:

```
<bean id="contentSource" class="com.endeca.infront.content.source.WorkbenchCon¬
tentSource"
          scope="singleton" init-method="init" destroy-method="destroy">
        <property name="storeFactory" ref="${store.factory}"/>
        <property name="defaultSiteRootPath" value="/pages" />
        <property name="appName" value="${workbench.app.name}"/>
    </bean>
```

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

Performance logging changes

The `com.endeca.infront.assembler.perf` package is deprecated in this release, with the exception of the `com.endeca.infront.assembler.perf.PerfEventFilter` class. For all other functionality, use the `com.endeca.infront.perf` and `com.endeca.infront.event` packages.

The `com.endeca.infront.assembler.perf.PerfEventFilter` is unchanged. By configuring it in your deployment descriptor file, `web.xml`, you can configure the HTTP request as the root event for performance logging:

```
<!-- The PerfFilter must go after the RequestContextFilter so it can have
access to the request -->
<filter>
    <filter-name>PerfFilter</filter-name>
   <filter-class>com.endeca.infront.assembler.perf.PerfEventFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>PerfFilter</filter-name>
    <url-pattern>*</url-pattern>
    <dispatcher>FORWARD</dispatcher>
    <dispatcher>REQUEST</dispatcher>
</filter-mapping>
```

The Dynamic Slot cartridge

The cartridge handler and configuration model for the Dynamic Slot cartridge have been modified with the addition of content folders to Tools and Frameworks 11.0.

The ContentSlotConfig configuration model

The `contentCollection` property of the `ContentSlotConfig` class is deprecated in this release. While implementations that include this property will continue to function, new dynamic slots should be configured with a `contentPaths` property, as well as template type and ID restrictions.

The ContentSlotHandler

The `ContentSlotHandler` includes a new `contentSource` property. This property specifies a `Con¬ tentSource` object that resolves the content paths on the input `ContentSlotConfig` object, and returns the corresponding content items. If you do not include it, the cartridge handler will not function.

The cartridge handler configuration is shown below in the Assembler context file for the Discover Electronics reference application:

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

Configuring dynamic content for the Guided Search Service

For each dynamic slot that you wish to populate as part of the response from the Guided Search Service, you must configure a `ContentSlotConfig` object. Each of these objects is set as a property of the default input content item for the `ContentSlotHandler`.

Specify the following properties for each instance of `ContentSlotConfig`:

Property name	Value
`contentPaths`	A List of String typed paths to the content folders from which you want to return results.
`templateTypes`	(Optional) A `List` of `String` typed template type restrictions for the dynamic slot.
`templateIds`	(Optional) A `List` of `String` typed template ID restrictions for the dynamic slot.
`ruleLimit`	The maximum number of content items to return from this collection. The Assembler returns up to this number of items that match the trigger criteria, based on priority.

Note: The content within a folder depends on the template type or ID restrictions configured for that folder in Experience Manager. While it is possible to configure your default `ContentSlotConfig` objects with any restrictions you wish, you should ensure that the type and ID restrictions you enter match those in Experience Manager. For example, it is possible to create a `ContentSlotConfig` object that is restricted by template type "MainContent," while the `contentPaths` property points to folders in Experience Manager that are restricted to "SecondaryContent" (and thus will never contain any "MainContent" content items).

Example

In the example below, the input content item to the `ContentSlotHandler` is a `ContentSlotListConfig` object. It is instantiated as "contentSlotList," and contains a `ContentSlotConfig` object for each dynamic

slot in the application. The `contentSlotList` is passed in to the `ConfigInitializer` that instantiates it as the input content item for the cartridge handler.

The `contentSlotList` for the Discover Electronics reference application is configured in the `Cartridge¬Handler_ContentSlotList` bean in the Spring context file, `assembler-context.xml`. For each content folder that is enabled for the Guided Search Service, a `ContentSlotConfig` bean appears in the `con¬tentSlotList` as in the example below:

```
<bean id="CartridgeHandler_ContentSlotList" class="com.endeca.infront.cartridge.Con¬
tentSlotListHandler"
 scope="prototype">
    <property name="contentItemInitializer">
        <bean class="com.endeca.infront.cartridge.ConfigInitializer" scope="re¬
quest">
            <property name="defaults">
                <bean class="com.endeca.infront.cartridge.ContentSlotListConfig"
 scope="singleton">
                    <property name="contentSlotList">
                        <list>
                          <bean class="com.endeca.infront.cartridge.ContentSlotCon¬
fig" scope="singleton">
                                <property name="contentPaths">
                                  <list>
                                   <value>/content/Right Column Spotlights</value>

                                  </list>
                                </property>
                                <property name="templateTypes">
                                  <list>
                                    <value>SecondaryContent</value>
                                  </list>
                                </property>
                                <property name="templateIds">
                                  <list>
                                    <value>RecordSpotlight</value>
                                    <value>RichTextSecondary</value>
                                  </list>
                                </property>
                                <property name="ruleLimit" value="3"/>
                            </bean>
                        </list>
                    </property>
                </bean>
            </property>
        </bean>
    </property>
</bean>
```

For detailed information on the `ContentSlotConfig` configuration model and its included properties, see the *Assembler API Reference (Javadoc)*.

Handling the Guided Search Service response

The Assembler returns the matching content items for each configured `ContentSlotConfig`, so the response consists of a list of lists of content items:

- `ContentSlotList` response content item
  - 1st content item, returned from a `ContentSlotConfig` with a `ruleLimit` of 3
    - Highest priority matching content item

    - Second highest priority matching content item
    - Third highest priority matching content item
- 2nd content item, returned from a `ContentSlotConfig` with a `ruleLimit` of 2
    - Highest priority matching content item
    - Second highest priority matching content item

Note that the Guided Search Service response is not view-friendly. You must parse it in your application logic to determine if any of the content items returned in the tree correspond to page sections you wish to populate for the end user's current location in the application.

Below is a sample JSON response from the Guided Search Service in the Discover Electronics reference application when the user selects the "Cameras" category:

```
"zones": {
    "@type": "ContentSlotList",
    "contentSlotList": [
        {
            "@type": "ContentSlot",
            "templateTypes": [
                "RecordSpotlight"
            ],
            "contents": [
                {
                    "@type": "RecordSpotlight",
                    "title": "Most Popular Cameras",
                    "name": "Spotlight Records",
                    "records": [
                        { … },
                        { … },
                        { … },
                        { … }
                    ]
                },
                {
                    "@type": "RecordSpotlight",
                    "title": "Top Rated Products",
                    "name": "Spotlight Records",
                    "records": [
                        { … },
                        { … },
                        { … }
                    ]
                }
            ],
            "contentPaths": [
                "/content/Right Column Spotlights"
            ],
            "ruleLimit": 3,
            "templateIds": [ ]
        }
    ]
},
```

It populates two sidebar Record Spotlight cartridges, the first with four records, and the second with three.

Content promotion changes

In previous releases of Tools and Frameworks, there was only one method for promoting content: the direct method. In 11.0, there are now two methods of promotion: direct and file-based.

Direct promotion should typically be used *within* your staging environment, while file-based promotion should be used *between* your staging and production environments.

Even if you plan on continuing to use the direct method, you must specify the promotion method to use in your Assembler configuration. In the reference implementations, you specify this method in the `assembler.properties` file by specifying a store factory. Store factories retrieve and store content and configuration specific to the application. There are two store types, but only one store would be used in any Assembler configuration.

- ecrStoreFactory - store for direct promotion used in a staging environment
- fileStoreFactory - store for file-based promotion used in a production environment.

You must also define the properties of the store that you choose in the `assembler-context.xml` file.

Note: The Assembler implementation included with Tools and Frameworks is configured through Spring. This guide assumes an application based around the included Assembler implementations. You can provide your own implementation if you wish to use an alternate means of configuring the Assembler.

1. Navigate to the Assembler properties file for you application, `WEB-INF/assembler.properties`.
2. Open `assembler.properties` in a text editor.
3. Edit the `store.factory` settings.

   For example, in the production environment for the Discover Electronics application, the store setting is `fileStoreFactory`:

   ```
   preview.enabled=false
   store.factory=fileStoreFactory
   user.state.ref=liveUserState
   media.sources.ref=liveMediaSources
   repository.configuration.path=./repository/${workbench.app.name}
   ```

   The `repository.configuration.path` is the location where content is extracted to when you promote content.
4. Save and close the file.
5. Navigate to the Assembler context file for you application, `WEB-INF/assembler-context.xml`
6. Open `assembler-context.xml` and find the Administration services section of the file.

   For example, in the production environment for the Discover Electronics application, the section looks like the following:

   ```
   <!--
          ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

           ~ Administration Service.
       -->
      <bean id="adminService" class="com.endeca.infront.assembler.servlet.admin.Ad¬
   ministrationService">
           <property name="storeFactory" ref="${store.factory}"/>
       </bean>
   ```

7. If you are using the file-based method of promotion, scroll down, and edit the properties in the `fileStoreFac¬tory` bean with values that are appropriate for your site.

For example, the `configurationPath` should be the the location where content is extracted to when you promote content.

```
<bean id="fileStoreFactory" class="com.endeca.infront.content.source.FileStoreFac¬
tory"
          init-method="init">
        <property name="configurationPath" value="${repository.configura¬
tion.path}"/>
    </bean>
```

8. If you are using the direct method of promotion, edit the `ecrStoreFactory` bean with values appropriate for your application:

```
<bean id="ecrStoreFactory" class="com.endeca.infront.content.source.EcrStoreFac¬
tory"
          init-method="init" destroy-method="destroy">
        <property name="isAuthoring" value="true"/>
        <property name="appName" value="${workbench.app.name}" />
        <property name="host" value="${workbench.host}" />
       <property name="clientPort" value="${workbench.publishing.clientPort}"/>

       <property name="serverPort" value="${workbench.publishing.serverPort}"/>

    </bean>
```

9. Save and close the `assembler-context.xml` file.
10. Restart Endeca Tools Services.

For detailed information on promoting content, see *Oracle Endeca Commerce Administrator's Guide*.

Migrating data applications based on the reference media data application

If you previously used the reference data application for indexing media assets in an MDEX Engine as a basis for your own data application, you should modify the crawl and included manipulators to correspond to those distributed in the Tools and Frameworks 11.0 package.

The CAS crawl for the reference data application has been modified to crawl media assets on a file system instead of a Java Content Repository. The updated crawl also uses a different set of record manipulators. See the *Assembler Application Developer's Guide* for details.

Authenticating users in custom Workbench extensions

The SHA-256 algorithm has replaced MD5 for generating the hash used to authenticate users who access Workbench extensions.

If you have existing Workbench extensions that authenticate users by verifying an MD5 hash, you must update your code to instead generate a hash using SHA-256.

For example, consider the following code:

```
// These values depend on what you defined in ws-extensions.xml
String extensionSecret="secret!@#$%^*(987654321";
final String authTokenParameterName = "auth";
final String timeStampParameterName = "timestamp";
```

```
// Set the tolerance, in milliseconds, before a request is considered too old
int allowedTimeStampSlackInMS = 5 * 60 * 1000;

// Calculate the hash of the substring of the URL and the shared secret
String url = request.getRequestURI() + "?" + request.getQueryString();
String findAuthToken = "&" + authTokenParameterName + "=";
url = url.substring(0, url.indexOf(findAuthToken) + findAuthToken.length());
String authCode = request.getParameter(authTokenParameterName);

MessageDigest md = MessageDigest.getInstance("MD5");
byte[] md5Hash = md.digest((url + extensionSecret).getBytes("UTF-8"));

StringBuffer hashCode = new StringBuffer();

for(int i : md5Hash) {
  String str = Integer.toHexString(i+128);
  if (str.length() < 2) {
    str = "0" + str;
  }
  hashCode.append(str);
}

// Compare the hash to the value of the AUTH token
if (!hashCode.toString().equals(authCode)) {
 // Authentication fails because AUTH token did not match
}

// Compare the time stamp of the request to the current time stamp
long currentTime = new Date().getTime();
long ts = Long.parseLong(request.getParameter(timeStampParameterName));

if ( Math.abs(ts - currentTime) > allowedTimeStampSlackInMS) {
 // Authentication fails because request is too old
}
```

The bolded section should be changed to something similar to this:

```
MessageDigest md = MessageDigest.getInstance("SHA-256");
byte[] secretDigest = md.digest((urlForHash + extensionSecret).getBytes("UTF-8"));

StringBuffer hashCode = new StringBuffer();

for (int i : secretDigest) {
  ...
}
```

Chapter 4

Behavioral Changes

This section describes changes in the 11.0 release that do not require action on your part, but do have an effect on how your Endeca application behaves after you upgrade.

Oracle Endeca Workbench

The following changes have been made to Workbench:

Locales

In previous releases of Tools and Frameworks, Workbench had a global locale setting which was configured in `webstudio.properties`:

```
# Localization
com.endeca.webstudio.locale=en_US
```

This locale applied to every user of Workbench, regardless of any preferred locale that the user's browser specified.

In the Tools and Frameworks 11.0 release, you can set a locale for users in the **User Management** interface of Workbench. If no locale is specified, then Workbench checks the user's browser settings for preferred locales that are supported in Workbench, and the first one that matches is used. If no match is found then the locale setting in `webstudio.properties` is used.

Assembler

The following changes have been made to Assembler cartridges:

Templates

In previous releases, templates were stored flat as XML files in a single directory on-disk: `config/cartridge_templates/` for a standard application. Each template had a unique file name and a template ID within the XML file. In Tools and Frameworks 11.0, each template is named `template.xml` and stored in a uniquely named directory. This directory name is now the template identifier. For example, in `config/cartridge_templates/mycustomcartidge/template.xml`, `mycustomcartridge` is the template identifier.

For the thumbnails in templates, if the URL begins with a protocol ("`http://`") or forward slash ("/"), the behavior will be as in previous releases. Any URL without a protocol or leading slash, however, is treated as relative to the root of the template structure. For example, the XML could simply have `<ThumbnailUrl>thumbnail.png</ThumbnailUrl>`.

In previous releases, application cartridges were stored in the Web Studio store. In this release, cartridges are stored in the ECR. The `set_template` script has been updated to automatically import cartridges into the ECR instead of the Web Studio store. The `get_template` script has similarly been updated.

Documentation changes

The following documents have been changed or removed in this release.

The *URL Optimization API for Java Developer's Guide*, *Cartridge Developer's Guide*, and *Experience Manager Editor Developer's Guide* have been combined with the content in the*Assembler Application Developer's Guide*.

Managing ATG user segments

In previous releases, using ATG user segments in Workbench required re-creating each segment. In the 11.0 release, you can configure Workbench to communicate with the ATG user segment server directly.

Do not manually create duplicates of the ATG user segments in Workbench, as this results in double entries that are unclear to content administrators.

Index