Oracle Endeca Commerce

Deployment Template Module for Product Catalog Integration Usage Guide

Version: 3.1.0
July 2012

The Deployment Template Module for Product Catalog Integration is the Endeca application template that integrates data records, dimension values, precedence rules, and schema information from product catalog systems into Endeca search applications.

Copyright and Disclaimer

Contents

About This Guide

This guide describes how to install, configure, and run the Deployment Template Module for Product Catalog Integration that is installed as part of Oracle Endeca Commerce Tools and Frameworks 3.1.0. The module integrates data from product catalog systems, such as Oracle ATG Web Commerce, with Endeca search applications.

These tasks are data integration tasks that are part of the back-end development of an Endeca application. This guide does not cover any aspects of front-end application development.

Who Should Use This Guide

This information is intended for experienced Endeca application developers.

Overview

The Deployment Template Module for Product Catalog Integration module provides capabilities to do the following:

- Create an Endeca application in a scripted manner.
- Create the required CAS Record Store instances for product catalog data records, dimension values, precedence rules, and schema information.
- Add supplemental dimension values, precedence rules, and schema information through the use of CSV files.

The module also provides a test data set that includes a small set of data records, dimension values, precedence rules, and schema information in XML format. You can examine the test data itself and then deploy the product catalog test application to better understand the application workflow and processing steps.

Required Endeca Software

The following Endeca software must be installed on the host where you install Tools and Frameworks:

- MDEX Engine
- Platform Services
- Tools and Frameworks
- Content Acquisition System
- Optionally, Developer Studio

See the *Oracle Endeca Commerce Compatibility Matrix* for version compatibility.

High-level Workflow to Integrate Product Catalog Systems and Endeca

The high-level workflow to integrate a product catalog system with Endeca consists of the following steps:

1. Deploy a product catalog application to familiarize yourself with the deployment process, test data, and application configuration. This module uses a version of the Discover Electronics reference application as a sample product catalog application.

2. Re-deploy a product catalog application and configure it for use with a product catalog system, for example, Oracle ATG Web Commerce.
3. Configure the product catalog system for use with Endeca. This step maps product catalog data to become Endeca records, dimension values, and schema after the data is exported. If you are using Oracle ATG Web Commerce, this task is described in the Oracle ATG Web Commerce documentation.
4. Export data records, dimension values, precedence rules, and schema information. If you are using Oracle ATG Web Commerce, this task is described in the Oracle ATG Web Commerce documentation.
5. If necessary, supplement the exported data with additional dimension values, precedence rules, and schema information by modifying CSV files.
6. Run a baseline update to process the data and CSV files.
7. Examine the indexed data in an Endeca reference application.

Using the Product Catalog Deployment Template Application

As mentioned in the overview, the main tasks are deploying the application and getting familiar with the deployment process and the test data. Then you configure the application to use production data from a product catalog system. Lastly, there are optional tasks to extend the application by adding additional language support.

Location of the Product Catalog Deployment Template Application

Remember that this module uses a version of the Discover Electronics reference application as the product catalog deployment template application.

After installing Tools and Frameworks, you can find the Deployment Template Module for Product Catalog Integration in the following location:

- `C:\Endeca\ToolsAndFrameworks\<version>\reference\discover-data-pci` on Windows.
- `/usr/local/endeca/ToolsAndFrameworks/<version>/reference/discover-data-pci` on UNIX.

Deploying the Product Catalog Application

There are two ways to deploy the product catalog application:

- Interactive deployment - you respond to command line prompts during the deployment process.
- Automated deployment - you run the deployment script and specify a deployment configuration file with values for the prompts.

To deploy the product catalog application:

1. Open a command prompt or command shell.

2. Navigate to the `<install path>\ToolsAndFrameworks\<version>\deployment_template\bin` directory on Windows, or to `<install path>/ToolsAndFrameworks/<version>/deployment-template/bin` on UNIX.

For example:

```
cd C:\Endeca\ToolsAndFrameworks\3.1.0\deployment_template\bin
```

3. Create the product catalog application in one of the following ways:

   A. Interactively, by running the deploy script and specifying the application descriptor file. For example, on Windows this command is the following:

   ```
   deploy.bat --app C:\Endeca\ToolsAndFrameworks\3.1.0\reference\
   discover-data-pci\deploy.xml
   ```

   Notes: Accept most defaults. Choose "Y" to install a base deployment. Specify `Discover` as the application name. Specify the path to CAS path using forward slashes (UNIX style). Specify the appropriate ISO language code, for example, `en` for English.

   B. In an automated way, by running the deploy script and specifying the deployment configuration file. For settings within this file, see the following topic "Options for the Deployment Configuration File."

   For example, on Windows this command is the following:

   ```
   deploy.bat --no-prompt --install-config
   C:\Endeca\ToolsAndFrameworks\3.1.0\reference\discover-data-
   pci\install-config-example.xml
   ```

4. Navigate to the `control` directory of the newly created application. For example, on Windows this command is the following:

   ```
   cd c:\Endeca\apps\<application name>\control
   ```

5. Run the `initialize_services` script.

The `initialize_services` script creates the Record Store instances for product data, dimension values, precedence rules, and schema information with the names below. The Record Store instances are prefixed with the application name and language code specified in the deployment descriptor file. In this case, the application name is `Discover` and the language code is `en`:

- `Discover_en_schema`
- `Discover_en_dimvals`
- `Discover_en_prules`
- `Discover_en_data`

6. Run the `load_baseline_test_data` script. The script loads 5684 records.

The script loads test data records, dimension value records, precedence rules, and schema.

This script also copies the CSV files to augment precedence rules, dimensions, and schema from `Discover\test_data\config_api_input` to `Discover\config\api_input`.

7. Run the `baseline_update` script.

8. Examine the indexed data in an Endeca front-end application. For example, start a Web browser and open the JSP reference application at `http://localhost:8006/endeca_jspref`.

Note: The URL assumes the reference application is running on the default Workbench port of 8006 and on localhost.

You should see 5684 product records.

9. Run the `load_partial_test_data` script to load additional records.

10. Run the `partial_update` script to process the partial test data.

11. Examine the indexed data in an Endeca front-end application. For example, start a Web browser and open the JSP ref application at `http://localhost:8006/endeca_jspref`.

Note: The URL assumes the reference application is running on the default Workbench port of 8006 and on localhost.

You should see 5685 product records.

Options for the Deployment Configuration File

You can modify the `install-config-example.xml` configuration file to set the application name, language, hostnames, and ports. This is useful if you are deploying multiple applications and you want customized default settings.

Oracle recommends that you make a copy of `<installation path>\ToolsAndFrameworks\3.1.0\reference\discover-data-pci\install-config-example.xml` for an application and change the file name to include the ISO language code for your data at the end of the file name. For example, if you were exporting German data, you would add `de` to the name: `install-config-myapp-de.xml`.

Use absolute paths for the deployment path, the path to deploy.xml, and for the CAS root.

For example:

```
<install app-name="Discover">
     <deployment-path>C:/Endeca/Apps</deployment-path>
     <base-module type="dgraph" />
     <additional-module type="custom">
           C:/Endeca/ToolsAndFrameworks/3.1.0/reference/discover-data-
pci/deploy.xml
     </additional-module>
     <options>
           <option name="eac-port">8888</option>
           <option name="workbench-port">8006</option>
           <option name="dgraph1Port">15000</option>
           <option name="authoringDgraphPort">15002</option>
           <option name="logserverPort">15010</option>
           <option name="casHost">localhost</option>
           <option name="casPort">8500</option>
           <option name="casRoot">C:/Endeca/CAS/3.0.2</option>
           <option name="casJarVersion">3.0.2</option>
           <option name="languageId">en</option>
     </options>
</install>
```

The deployment configuration file contains settings for the following values:

- **app-name**: Specifies the name of the Endeca application.
- **deployment-path**: Specifies the path where the Endeca applications are deployed.
- **base-module**: Specifies the absolute path to the deployment descriptor file (deploy.xml).
- **additional-module**: Specifies whether the application is a Dgraph or Agraph deployment.
- **eac-port**: Specifies the port of the local EAC instance.
- **workbench-port**: Specifies the port of Endeca Workbench.
- **dgraph1Port**: Specifies the port of the live Dgraph.
- **authoringDgraphPort**: Specifies the port of the authoring Dgraph.

- **logserverPort**: Specifies the port of the Log Server.
- **casHost**: Specifies the hostname of the Endeca CAS Service.
- **casPort**: Specifies the port of the CAS Server to use
- **casRoot**: Specifies the path to the Endeca Content Acquisition System (CAS). Ensure the CAS path uses forward slashes (UNIX style).
- **casJarVersion**: Specifies version number of the CAS JAR.
- **languageId**: Specifies the ISO code for the language of the Endeca application.

If more complex configurations are required, Oracle recommends that you extend the deployment template module to add more tokens (such as more Dgraph ports or Dgraph hosts) to the `deploy.xml`, `install-config.xml`, `environment.properties` and `AppConfig.xml`. For details, see "Custom application descriptors" in the Deployment Template Usage Guide.

This ensures that if you add a new language, the same configuration pattern as the other languages is used.

Note: In the options file, the language code is hardcoded in the application name as part of the naming convention. This is to help distinguish multiple applications in an Endeca Workbench instance.

Configuring an Application to use Production Data from a Product Catalog System

After you have familiarized yourself with the product catalog application, you can re-deploy another instance of the application and configure it for use with production data from a product catalog system.

The processing model of a production application is very similar to an application configured to use test data. However, there are a few key differences:

- Instead of running the `load_baseline_test_data` script, you export production data, dimension information, precedence rules, and schema information directly to Endeca Record Store instances that were created by the `initialize_services` script.
- You supplement the data with CSV files and store the CSV files in `<application name>\config\api_input` rather than `<application name>\test_data\config_api_input.`

To configure an application to use production data:

1. Open a command prompt or command shell.

2. Navigate to the `<install path>\ToolsAndFrameworks\<version>\deployment_template\bin` directory on Windows, or to `<install path>/ToolsAndFrameworks/<version>/deployment-template/bin` on UNIX. For example:

```
cd C:\Endeca\ToolsAndFrameworks\3.1.0\deployment_template\bin
```

3. Create an application in one of the following ways:

   A. Interactively, by running the deploy script and specifying the application descriptor file. For example, on Windows this command is the following:

```
deploy.bat --app C:\Endeca\ToolsAndFrameworks\3.1.0\reference
\discover-data-pci\deploy.xml
```

Notes: Accept most defaults. Choose "Y" to install a base deployment. Specify an application name. Specify the path to CAS path using forward slashes (UNIX style). Specify the appropriate ISO language code (`en`).

B. In an automated way, by running the deploy script and specifying the deployment configuration file. For example, on Windows this command is the following:

```
deploy.bat --no-prompt --install-config
C:\Endeca\ToolsAndFrameworks\3.1.0\reference\discover-data-
pci\install-config-example.xml
```

For settings within this file, see "Options for the Deployment Configuration File."

4. On the file system, navigate to `<application name>\config\media` and delete all the images that are used in the sample data set.

5. On the file system, navigate to `<application name>\config\ifcr` and delete all the files that are used in the sample data set for merchandising configuration.

6. On the file system, navigate to `<application name>\control` and open the `initialize_services` script in a text editor.

7. Delete calls to import sample content and calls to import sample media.

8. From a Command Prompt window, navigate to the `control` directory of the new application. For example, on Windows this command is the following:

```
cd c:\Endeca\apps\<application name>\control
```

9. Run the `initialize_services` script.

The `initialize_services` script creates the Record Store instances for product data, dimension values, precedence rules, and schema information with the following names:

- `<application name>_en_schema`
- `<application name>_en_dimvals`
- `<application name>_en_prules`
- `<application name>_en_data`

10. Export data records, dimension values, precedence rules, and schema information from your product catalog system.

- If you are exporting from Oracle ATG Web Commerce, this task is described in the Oracle ATG Web Commerce documentation.
- If you are using another product catalog system, ask your vendor if they provide an Endeca export module.
- Alternatively, you can export data to Record Store instances by using any of the following approaches:
    - Programmatic export via the Record Store API.
    - Export/load via the Record Store Command-line Utility.
    - Create a CAS crawl that crawls the product catalog and writes to the Record Store instances.
    - Create a Forge pipeline configured for Record Store output

For more information and record property requirements, see "Record Properties for Endeca Records".

11. If it is necessary to supplement the exported data by creating precedence rules, modifying dimension values, or modifying schema information, do the following:

    A. Open the CSV files in `<application name>\config\api_input\`.

    B. Modify the files as described in "Managing Precedence Rules", "Managing Dimension Values", or "Managing Schema Information" below.

12. Run the `baseline_update` script.

13. Examine the indexed data in an Endeca front-end application such as the Endeca JSP Reference Application.

14. As necessary, run the `partial_update` script and examine the indexed data.

Modifying an Instance Configuration Using Developer Studio

In some cases, you may need to modify the instance configuration XML files stored in `<application name>\config\pipeline` directory before you run a `baseline_update` script or a `partial_update` script. For example, it may be necessary to adjust search interface settings, relevance ranking modules, and so on.

To modify an instance configuration using Developer Studio: an application to use production data:

1. From the Windows Start menu, select All Programs > Endeca > Developer Studio > Developer Studio 6.1.3.

2. From the File Menu, select Open Project…

3. Browse to the `<application name>\config\pipeline` directory and select your project's .esp file.

4. Modify the Developer Studio project (i.e. the instance configuration) as necessary.

Record Properties for Endeca Records

This topic describes the record requirements for exporting data using the Record Store API or the Record Store Command-line Utility. This information in this topic is not necessary if you are integrating Endeca with Oracle ATG Web Commerce.

The `initialize_services` script creates the following record store instances:

- `<application name>_en_data` – contains data records
- `<application name>_en_dimvals` – contains dimension value records
- `<application name>_en_prules` – contains precedence rule records
- `<application name>_en_schema` – contains schema records

If you are writing to these record store instances using the Record Store API or the Record Store Command-line Utility, the records must contain name/value properties that are the same as the properties listed in the corresponding CSV files. (The CSV file itself is effectively the record schema.)

For data records, ensure that each record contains a `record.spec` property that specifies a unique Id for each record in this record feed.

For dimension value records, see "Format of the Dimension Values CSV File". In addition, add a property named `dimval.qualified_spec` that specifies a unique Id for each record in this record feed.

For precedence rule records, see "Format of the Precedence Rule CSV File". In addition, add a property named `precedence.qualified_spec` that specifies a unique Id for each record in this record feed.

For properties in schema records, see "Format of the Schema CSV File".

Adding a New Language to the Module

A typical Endeca application contains one language per MDEX Engine. When product catalog data in multiple languages must be exported, you should deploy a new Endeca application with a separate set of Record Store instances, Dgraphs, and a log server for each language.

The steps to do this are similar to those for the initial deployment except that you specify a *languageId* value that corresponds to the language of the source data.

To add a new language:

1. Navigate to the installation directory for the module. For example, on Windows this command is the following:

```
cd C:\Endeca\ToolsAndFrameworks\3.1.0\reference\discover-data-pci
```

2. Make a copy of the default deployment configuration file (`install-config-example.xml`), and add the ISO language code to the end of the file name. For example, if you were exporting German data, you would add de to the name: `install-config-myapp-de.xml`

3. Edit the new deployment configuration file to set the application settings and be sure to set the `languageId` with the correct ISO language code. This code matches the informal naming convention used in the `app-name` setting. For example:

```
<install app-name="myappde">
       <deployment-path>C:/Endeca/Apps</deployment-path>
       <base-module type="dgraph" />
       <additional-module type="custom">
              C:/Endeca/ToolsAndFrameworks/3.1.0/reference/discover-data-
pci/deploy.xml
       </additional-module>
       <options>
              <option name="eac-port">8888</option>
              <option name="workbench-port">8006</option>
              <option name="dgraph1Port">15000</option>
              <option name="authoringDgraphPort">15002</option>
              <option name="logserverPort">15010</option>
              <option name="casHost">localhost</option>
              <option name="casPort">8500</option>
              <option name="casRoot">C:/Endeca/CAS/3.0.2</option>
              <option name="casJarVersion">3.0.2</option>
              <option name="languageId">en</option>
       </options>
</install>
```

4. Navigate to the `<install path>\ToolsAndFrameworks\<version>\deployment_template\bin` directory on Windows, or to `<install path>/ToolsAndFrameworks/<version>/deployment-template/bin` on UNIX. For example, on Windows this command is the following:

```
cd C:\Endeca\ToolsAndFrameworks\3.1.0\deployment-template\bin
```

5. Deploy the application using the new deployment configuration file. For example, on Windows this command is the following:

```
deploy.bat --no-prompt --install-config
C:\Endeca\ToolsAndFrameworks\3.1.0\reference\discover-data-pci
\install-config-myapp-de.xml
```

6. At this point, you can run the `initialize_services` script and the `baseline_update` script as you normally would.

Managing Product Catalog Data in Endeca

After exporting production data from a product catalog system, you can add additional data records or supplement the data by modifying CSV files. The update scripts process the CSV files.

Managing Precedence Rules

Precedence rules control when certain dimensions for a given navigation state are displayed.

Rules can be exported from a product catalog system to a record store instance, and, if necessary, you can create additional rules by adding them to `precedence_rules.csv`. This file is processed during a baseline update.

When you deploy a product catalog application, the deployment script creates `precedence_rules.csv` in `<application name>\config\api_input`.

To manage precedence rules:

1. Open `precedence_rules.csv` in a text editor or Microsoft Excel.

2. Modify the CSV file with your custom precedence rule changes. See the “Format of the Precedence Rules CSV File” below for details about the column names and values.

3. Save and close the file.

Note: When using Microsoft Excel, ensure that the file is saved as CSV (plain text) and not in a binary format like .xls or .xlsx. Also ensure that the line endings are in Windows format (\r\n) and not UNIX (\n).

4. If you are using the test application, run the `load_baseline_test_data` script, or if you are using production data from a product catalog system, export the data to CAS Record Store instances.

5. Run a baseline update. The update processes the `precedence_rules.csv` file and all data contained in Record Store instances.

6. Examine the indexed data in an Endeca front-end application such as the Endeca JSP Reference Application.

Notes about processing precedence rules:

- Precedence rules with missing required properties are logged and skipped: no exception is thrown.
- Precedence rules with missing or unrecognized types are logged and changed to "standard" type.

Format of the Precedence Rules CSV File

The format of the file is plain text and comma delimited. The columns in the file are defined as follows:

- **precedence.source_dimension_name**: Optional. Specifies the spec of the source dimension as defined in the schema Record Store instance for the application.

- **precedence.source_dimval_spec**: Optional. Specifies the spec of the dimension value of the source dimension as defined in the dimension value Record Store instance for the application.
- **precedence.target_dimension_name**: Optional. Specifies the name of the dimension to show as defined in the schema.
- **precedence.type**: Optional. Specifies which type this rule is of (standard or leaf). The default value is standard.

Managing Dimension Values

If necessary, you can add additional dimension values to an Endeca application by defining them in a CSV file and processing the file during a baseline update. This approach is useful if there are dimension values that do not exist in the product catalog system, and therefore have to be created manually, or if there are dimension values that may be complicated to export, such as range dimension values.

When you deploy a product catalog application, the deployment script creates `dimension_values.csv` in `<application name>\config\api_input`.

To manage dimension values:

1. Open `dimension_values.csv` in a text editor or Microsoft Excel.

2. Modify the CSV file with your custom dimension value changes. See "Format of the Dimension Values CSV File" below for details about the column names and values.

3. Save and close the file.

Note: When using Microsoft Excel, ensure that the file is saved as CSV (plain text) and not in a binary format like .xls or .xlsx. Also ensure that the line endings are in Windows format (\r\n) and not UNIX (\n).

4. If you are using the test application, run the `load_baseline_test_data` script, or if you are using production data, export the data to CAS Record Store instances.

5. Run a baseline update. The update processes `dimension_values.csv` and all data contained in Record Store instances.

6. Examine the indexed data in an Endeca front-end application such as the Endeca JSP Reference Application.

Format of the Dimension Values CSV File

The format of the file is plain text and comma delimited. The columns in the file are defined as follows:

- **dimval.dimension_spec:** Required. Specifies the specification of the dimension name as defined in the Record Store instance or schema.csv file.
- **dimval.spec:** Required. Specifies the specification of the dimension value. If this attribute is a root dimension value, it should be set to **dimval.dimension_spec**.
- **dimval.display_name:** Optional. Specifies the name to display for this dimension value when rendered on a result page. The default value is the value of dimval.spec.
- **dimval.parent_spec:** Optional. Specifies the parent dimension value for hierarchical dimensions.
- **dimval.range.lower_bound:** Optional. Specifies a numeric value for the lower bound of the range. Leave empty to for a range dimension with "everything below 100 Euros"

- **dimval.range.lower_bound_inclusive:** Required if **dimval.range.lower_bound** is defined. Specifies a Boolean value to indicate if the lower bound value should be included in the range. There is no default value.
- **dimval.range.upper_bound:** Optional. Specifies a numeric value for the upper bound of the range. Leave empty to for a range dimension with “everything above 10000 Euros”
- **dimval.range.upper_bound_inclusive:** Required if **dimval.range.upper_bound** is defined. Specifies a Boolean value to indicate if the upper bound value should be included in the range. There is no default value.
- **dimval.display_order:** Optional. A numeric value that specifies the display order of dimension values. Dimension values with a display order defined are sorted lower to higher, and dimension values with no display order are sorted alphabetically.
- **dimval.classify_synonym:** Optional. Specifies a synonym value that is mapped to the dimension value.
- **dimval.search_synonym:** Optional. Specifies a synonym value for the dimension value. If specified, a dimension value search for the synonym value returns the dimension value.
- **dimval.match_type:** Optional. A value that specifies how dimension values are matched. Valid values are "exact", "range", or "sift". The default value is "exact".
- **dimval.range.comparison_type:** Optional. A value that specifies how range dimensions are matched. Valid values are "string", "decimal", or "integer". The default value is "string".
- **dimval.prop.*:** Optional. Any number of dimension value properties can be attached. The prefix must be: dimval.prop. For example, specifying "dimval.prop.currency" results in a key property named "currency" with a value equal to the value of dimval.prop.currency (e.g. "USD").

Managing Schema Information

Schema information describes the search configuration settings you can make for Endeca properties, dimension values, search interfaces, and so on. You can update these settings by modifying `schema.csv`.

When you deploy a test application, the deployment script creates `schema.csv` in `<application name>\config\api_input`.

Schema properties within this file have the following format and scope:

- **attribute.<description>** - These schema settings are common to both Endeca properties and dimensions.
- **attribute.property.<description> -** These schema settings apply to Endeca properties.
- **attribute.dimension.<description> -** These schema settings apply to Endeca dimensions.

Any changes you make using Developer Studio will override any schema values in this file or in a Record Store instance.

To manage schema information:

1. Open `schema.csv` in a text editor or Microsoft Excel.

2. Modify the CSV file with your custom changes. See the topic below for details about the column names and values.

3. Save and close the file.

Note: When using Microsoft Excel, ensure that the file is saved as CSV (plain text) and not in a binary format like .xls or .xlsx. Also ensure that the line endings are in Windows format (\r\n) and not UNIX (\n).

4. If you are using the test application, run the `load_baseline_test_data` script, or if you are using production data, export the data to CAS Record Store instances.

5. Run a baseline update. The update processes `schema.csv` and all data contained in Record Store instances.

6. Examine the indexed data in an Endeca front-end application such as the Endeca JSP Reference Application.

Format of the Schema CSV File

The format of the file is plain text and comma delimited. The columns in the file are defined as follows:

- **attribute.name**: Required. A string indicating the name of the property or dimension.
- **attribute.type**: Optional. A string specifying either "dimension" or "property" depending on what the attribute is representing. The default value is "property".
- **attribute.display_name**: Optional. A string indicating the name of the property or dimension to display when rendered on a result page. The default value is the value of attribute.name.
- **attribute.offline_sort**: Optional. A Boolean value indicating whether to optimize record sorting for this property or dimension. A value of `true` optimizes record sorting. The default value is `false`.
- **attribute.record_filterable**: Optional. A Boolean value indicating whether to enable recording filtering for this property or dimension. Record filtering presents a subset of the data to the end-user. A value of `true` enables the property or dimension for record filtering. The default value is `false`.
- **attribute.rollup_key**: Optional. A Boolean value that enables aggregated Endeca record creation by allowing rollups based on this property or dimension. The default value is `false`.
- **attribute.search_interface**: Optional. A pipe (|) separated list of search interfaces to which this attribute should be added as a member. The names must be pipe-delimited if specified in the CSV file, but when loading records into a Record Store instance, names may be either pipe-delimited or multi-valued.

  Notes about search interfaces:

  - If you are not using the default “All” search interface, you should create a search interface before you add additional member attributes using this schema property. This allows you to create an associated relevance ranking strategy when creating the search interface. Otherwise, search interfaces are created without an associated relevance ranking strategy
  - Also note that editing a search interface in Developer Studio requires at least one member attribute that is managed by Developer Studio. It may be necessary for you to create a placeholder member to add to the search interface. You can then populate the search interface with valid member attributes using the schema.csv file. (By default, the test application creates a dummy placeholder member attribute for the “All” search interface.)
  - In this release, you cannot modify the order of members within a search interface. By default, members are ordered alphabetically.

- **attribute.search.searchable**: Optional. A Boolean value that indicates whether the property or dimension is enabled for record search and added to the “All” search interface. For Endeca properties, record search finds all records in an Endeca application that have a property whose value matches a term that the user provides. For dimensions, record search finds all records in an Endeca application that have a dimension whose value matches a term the user provides. If unspecified, the attribute is not searchable. The default value is `false`.
- **attribute.search.wildcard**: Optional. A Boolean value that indicates the property or dimension is enabled for wildcard search. You must enable each property or dimension that you want available for wildcard searching. The default value is `false`.
- **attribute.show_with_list**: Optional. A Boolean value that indicates the property or dimension appears in the record list. The default value is `true`.
- **attribute.show_with_record**: Optional. A Boolean value that indicates the property or dimension displays on the record page. The default value is `true`.
- **attribute.source_name**: Optional. A string indicating the name or names of the property or dimension in the source data that map to this attribute. The names should be pipe-delimited if specified in a CSV file, but when loading records into a Record Store instance, names may be pipe-delimited or multi-valued. The default value is the value of attribute.name.
- **attribute.property.data_type**: Required if the attribute.type is property, otherwise ignored. Valid values are "string", "integer", "decimal", "datetime", or "geocode".
- **attribute.dimension.autogen**: Optional. A Boolean value that indicates whether the dimension has been autogenerated by Forge. The default value is `false`.
- **attribute.dimension.compute_refinement_counts**: Optional. A Boolean value that indicates whether refinement counts should be computed for this dimension. The default value is `true`.
- **attribute.dimension.dyn_ranking_enabled**: Optional. A Boolean value that enables dynamic dimension value ranking, which orders dimension values according to their frequency of appearance within the current record set (that is, the most popular dimension values are returned). The default value is `false`.
- **attribute.dimension.dyn_ranking_refinement_count**: Optional. A numeric value that specifies how many refines are returned. The default value is 10.
- **attribute.dimension.dyn_ranking_more**: Optional. A Boolean value that enables the more feature. The default value is `false`.
- **attribute.dimension.dyn_ranking_sort**: Optional. A value that defines the default sort order used for dimension values within this dimension. This sort order is used when no other types of dimension value sorting have been specified. Valid values are "alpha" or "refinement-count". The default value is “alpha”.
- **attribute.dimension.hidden**: Optional. A Boolean that specifies whether or not this dimension is shown in the navigation controls. A value of `true` hides the dimension value. The default value is `false`.
- **attribute.dimension.hierarchy_dim_search**: Optional. The default value is `false`.
- **attribute.dimension.hierarchy_record_search**: Optional. The default value is `false`.
- **attribute.dimension.multiselect_type**: Optional. Valid values are "none", multi-or", or "multi-and". The default value is “none”.

- **attribute.dimension.refinements_sort_order**: Optional. Specifies the sort order of dimension values. Valid values are "alpha" or "numeric". The default value is “alpha”.

Understanding the Priority Order of Attribute Values

There are three integration points in the Deployment Template Module for Product Catalog Integration that can set attribute values. These integration points include:

- A Forge Configuration Generation Adapter (a CADK adapter)
- CAS Record Store instances
- The `schema.csv` file

Each integration point can override attribute values, so it is important to understand the following priority order of how attribute values are set:

1. The Forge Configuration Generation Adapter in the Forge pipeline creates new attributes based on property and dimension templates. The templates specify default values for each attribute. (The adapter sets the lowest priority.)

2. Any attributes that a product catalog system exports to a Record Store instance override the default values set by the adapter. (Record Store instance data sets medium priority.)

3. Any attributes that you specify in `schema.csv` override the other two. (The schema file is the highest priority.)

Globally Modifying the Default Values for Schema Attributes

If necessary, you can modify the default values that the Forge Configuration Generation Adapter uses to create new attributes. This configuration provides a way to globally set the value of new schema attributes. (Remember that you can override schema attributes as necessary by changing them in `schema.csv`.)

To globally modify the default values:

1. Navigate to <application directory>\config\script.

2. Open `fcm.context.xml` in a text editor.

3. Scroll down until you locate the `propertyTemplate` and `dimensionTemplate` sections of the file.

4. Modify the `propertyTemplate` section to change the defaults for Endeca properties and modify `dimensionTemplate` section to change the defaults for Endeca dimension values. The properties are defined in "Format of the Schema CSV File".

5. Save and close the file.

For example, this snippet of XML shows the `propertyTemplate` and `dimensionTemplate` sections you modify:

```
  <!-- this encodes the property default values from the schema feed xls
spreadsheet -->
  <bean id="propertyTemplate"
class="com.endeca.itl.fcm.beans.PropertyConfiguration">
    <property name="language" value="${LANGUAGE_ID}" />
    <property name="isSearchable" value="false" />
    <property name="isWildcardSearchable" value="false" />
    <property name="isSortable" value="false" />
    <property name="isFilterable" value="false" />
    <property name="showOnListPage" value="true" />
```

```
    <property name="showOnDetailPage" value="true" />
    <property name="enableForRollup" value="false" />
  </bean>

  <bean id="dimensionTemplate"
class="com.endeca.itl.fcm.beans.DimensionConfiguration">
    <property name="language" value="${LANGUAGE_ID}" />
    <property name="isSearchable" value="false" />
    <property name="isWildcardSearchable" value="false" />
    <property name="isSortable" value="false" />
    <property name="showOnListPage" value="true" />
    <property name="showOnDetailPage" value="true" />
    <property name="enableForRollup" value="false" />

    <property name="refineSortType" value="ALPHA" />
    <property name="dimSearchHierarchy" value="false" />
    <property name="recSearchHierarchy" value="false" />
    <property name="refineShowRecordCounts" value="false" />
    <property name="multiSelect" value="NONE" />
    <property name="refineDynRank" value="false" />
    <property name="refineDynRankCount" value="10" />
    <property name="refineDynRankMore" value="false" />
    <property name="refineDynRankSortType" value="DEFAULT" />
    <property name="isHidden" value="false" />
    <property name="matchMode" value="NORMAL" />
  </bean>
```

The following table provides a mapping of properties in the `schema.csv` file to their equivalent property in `fcm.context.xml`.

Schema.csv property	fcm.context.xml property	Scope
attribute.search.searchable	isSearchable	propertyTemplate and dimensionTemplate
attribute.search.wildcard	isWildcardSearchable	propertyTemplate and dimensionTemplate
attribute.offline_sort	isSortable	propertyTemplate and dimensionTemplate
attribute.show_with_list	showOnListPage	propertyTemplate and dimensionTemplate
attribute.show_with_record	showOnDetailPage	propertyTemplate and dimensionTemplate
attribute.rollup_key	enableForRollup	propertyTemplate and dimensionTemplate
attribute.dimension.refinements_sort_order	refineSortType	dimensionTemplate only
attribute.dimension.hierarchy_dim_search	dimSearchHierarchy	dimensionTemplate only

attribute.dimension.hierarchy_record_search	recSearchHierarchy	dimensionTemplate only
attribute.dimension.compute_refinement_counts	refineShowRecordCounts	dimensionTemplate only
attribute.dimension.multiselect_type	multiSelect	dimensionTemplate only
attribute.dimension.dyn_ranking_enabled	refineDynRank	dimensionTemplate only
attribute.dimension.dyn_ranking_refinement_count	refineDynRankCount	dimensionTemplate only
attribute.dimension.dyn_ranking_more	refineDynRankMore	dimensionTemplate only
attribute.dimension.dyn_ranking_sort	refineDynRankSortType	dimensionTemplate only
attribute.dimension.hidden	isHidden	dimensionTemplate only

Tuning Performance Considerations

The export operation from a product catalog system can write large amounts of data to the CAS Record Store instances. In cases where performance becomes an issue, you may need to fine-tune the record store configuration, especially when running with large data volumes and high update frequencies. For details on cleaning a Record Store instance, see the *CAS Developer's Guide*.

Also, write operations to the Record Store instances may cause the CAS JVM to run out of memory. For details about increasing the amount of memory allocated to CAS, see the *CAS Developer's Guide*.