# Oracle® Enterprise Data Quality for Product Data

Endeca Connector Installation and User's Guide

Release 11g R1 (11.1.1.6)

**E29135-03**

March 2014

ORACLE

Oracle Enterprise Data Quality for Product Data Endeca Connector Installation and User's Guide, Release 11g R1 (11.1.1.6)

E29135-03

# Contents

## 4 Running the Endeca Connector



## 5 Logging and Tracing Data



## 6 Using the Endeca Connector Adapter



## A Performance



## B DSA Format



## C Endeca Connector Robustness



## D Setting Up an Example Endeca Connector Project

## E Endeca Connector Troubleshooting

# Preface

This purpose of this document is to describe integrating the Oracle Enterprise Data Quality for Product Data with the Endeca Information Access Platform.

You must have Oracle Enterprise Data Quality for Product Data 11g and Endeca 6.0 or later installed on your server.

This document is used after both of these systems have been installed and configured.

## Audience

You should have a basic understanding of the Endeca Information Access Platform application servers and client workbench.

You should have a basic understanding of Oracle Enterprise Data Quality for Product Data.

A thorough understanding of the material in this guide is required for the following customer personnel:

- Application and Solution Owners
- Business Analysts
- Information Technology Administrators
- Subject Matter Experts (SME)

## Documentation Accessibility

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.

### Access to Oracle Support

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.

## Related Documents

For more information, see the following documents in the Oracle Enterprise Data Quality for Product Data 11g documentation set:

- The *Oracle Enterprise Data Quality for Product Data Oracle DataLens Server Installation Guide* provides detailed Oracle DataLens Server installation instructions.
- The *Oracle Enterprise Data Quality for Product Data Oracle DataLens Server Administration Guide* provides information about managing an Oracle DataLens Server including users and user roles.
- The *Oracle Enterprise Data Quality for Product Data .NET Interface Guide* provides information about installing and using the Oracle DataLens Server .NET API.
- The *Oracle Enterprise Data Quality for Product Data Java Interface Guide* provides information about installing and using the Oracle DataLens Server Java APIs.
- The *Oracle Enterprise Data Quality for Product Data Application Studio Reference Guide* provides information about creating and maintaining Data Service Applications (DSAs).
- The *Oracle Enterprise Data Quality for Product Data AutoBuild Reference Guide* provides information about creating an initial data lens based on existing product information and data lens knowledge.
- The *Oracle Enterprise Data Quality for Product Data Knowledge Studio Reference Guide* provides information about creating and maintaining data lenses.
- The *Oracle Enterprise Data Quality for Product Data Governance Studio Reference Guide* provides information about building projects to analyze your transformed data, create reports to show the quality of your data, and identify missing attributes.
- The *Oracle Enterprise Data Quality for Product Data Services for Excel Reference Guide* provides information about creating a DSA based on data contained in a Microsoft Excel worksheet.

See the latest version of this and all documents listed at the Oracle Enterprise Data Quality for Product Data Documentation 11g web site at:

`http://docs.oracle.com/cd/E35636_01/index.htm`

For information about the Endeca Information Access Platform, see the Oracle MDEX Engine Documentation web site at

`http://docs.oracle.com/cd/E29584_01/index.htm`

# Conventions

The following text conventions are used in this document:

| Convention | Meaning |
|---|---|
| **boldface** | Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary. |
| *italic* | Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values. |
| `monospace` | Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, text that you enter, or a file, directory, or path name. |
| **`monospace`** | Boldface, monospace type indicates commands or text that you enter. |

# 1

# Overview

Poor data undermines website usability by degrading user experience leading to fewer sales. In contrast, good data supports the guided navigation that is important for website usability. The Oracle Enterprise Data Quality for Product Data (EDQP) Endeca Connector enables you to dramatically improve the guided navigation of your website and requires good data to operate efficiently. With good data, it is possible to experience the full richness of search using guided navigation, including dimension search with drill-down. Good data also eliminates large unsearchable result sets and redundant navigation options. The EDQP Endeca Connector allows you to create good data to support your Endeca implementation.

The Endeca Connector automates data standardization, reducing manual effort, increasing quality, reliability and scalability. The business maintains the rules independent of the Information Technology organization, enabling the business to make and deploy new changes very quickly. The business has increased flexibility to meet merchandising requirements, including more complete categories with better quality. As a result, more dimensions are available to improve the search experience. The Endeca Connector optimizes navigation, reduces cost, risk and time.

In production, Endeca resides on a server and responds to calls from web pages, providing guided navigation services. EDQP is used to extract and standardize the product data that Endeca uses. These two systems are connected by the Endeca Connector that is used by an Endeca pipeline at data-load time. This Endeca Connector adapter allows Endeca to call an EDQP Data Service Application (DSA) to process individual lines of data. This DSA uses EDQP elements to extract and return attributes (called dimensions by Endeca) based on the attributes defined in item definitions in one or more specific data lenses.

## Endeca Connector Components

There are two components to the Oracle Enterprise Data Quality for Product Data (EDQP) Endeca Connector as follows:

#### PDQ-Endeca Connector Attribute Discovery

This initialization component updates the Endeca Developers Studio project with Dimensions, Properties, and Precedence Rules directly from the DataLens Server.

#### PDQ-Endeca Connector Adapter

This run-time component is integrated with the Endeca Forge Processing and is used when processing the input data.

## PDQ-Endeca Connector Attribute Discovery

The PDQ-Endeca Connector program is run to define the Endeca Dimensions and to create the input data mapping for the dimensions in the pipeline configuration file. There are additional configuration options for setting the dimension properties for all the EDQP-generated dimensions. The PDQ-Endeca Connector Attribute Discovery configures the Endeca project directly from an Oracle DataLens Administration Server.

This component is implemented as a DSA Add-In Transformation and is completely integrated with the EDQP system.

The PDQ-Endeca Connector program should only be run when there have been changes to the data lenses to add or modify the item definitions. It should also be run if new data lenses are added to the main Process Map. Simply re-load the Endeca Project in the Developer Studio to see the changes made by the PDQ-Endeca Connector program.

## Endeca Connector Adapter

The Endeca Connector Adapter is a Java class that implements the Endeca application programing interface (API) for adding attributes into the data flow during the Endeca baseline update processing. The PDQ-Endeca Connector Adapter transforms data directly from a Production Oracle DataLens Server.

The PDQ-Endeca Connector Adapter is integrated into the Endeca pipeline process during the configuration, and does not need to be modified after installation. The PDQ-Endeca Connector Adapter runs whenever the Endeca baseline update is performed.

### Libraries

The following libraries comprise the PDQ-Endeca Connector Adapter:

| Library | Description |
|---|---|
| `opdq-api1.jar` | The library for the Oracle DataLens Server API. |
| `opdq-core.jar` | The library for the Oracle DataLens Server core and utility classes used by the API. |
| `jdom-1.0.jar` | The library for third party components. |
| `opdq-connector-endeca.jar` | The library for the PDQ-Endeca Connector DSA Transform Add-In and the PDQ-Endeca Connector Adapter. |

## Endeca Connector Processes

The initial part of the integration happens with the Endeca Loader process. The loader process is a DSA, which is configured to extract the relevant attributes from the data lenses and insert them into the Endeca pipeline as dimensions. The configuration for this is composed of two main pieces, the dimension discovery, and the precedence discovery.

Dimension discovery requires the following parameters: project name, project (file) location, and standardization name. Using the project location, the DSA will read in the `pipeline.epx` file and look for the PdqAdapter Java Manipulator. From this Java Manipulator, it retrieves the name of the parser DSA from the value of the `DSA_MAP` pass through variable. It then looks at the configuration of the parser DSA, and identifies all of the data lenses used for data processing in the DSA. The 'dlsapp_parser' DSA is included as an example in the connector package. The 'dlsapp_parser'

DSA package includes the 'Writing_Instruments' data lens. The data lens is then opened and the attributes scanned. The discovery looks for the standardization defined in the loader parameters then reads all the attributes used in the standardization (equivalent to what is displayed on the Order Attributes sub-tab in the Knowledge Studio.)

After the dimension discovery has identified t the attributes in the data lenses, they must be added to the pipeline. The discovery process directly modifies the Endeca XML files (and the `pipeline.epx`, which is really an XML file). Both are added to the dimension rule, as well as a mapping rule for properties of the same name. The loader typically is aware of what must be modified and attempts not to override anything that has been manually set. The loader can only scan one external dimensions file so if you have multiple x-defs, those dimensions can end up being duplicated. Since the loader tries not to override anything that has been manually configured, adding a property or dimension directly into the pipeline prevents the loader from creating conflicting external dimensions (and the parser then uses the existing dimension.)

The precedence discovery has a simpler configuration and process. The precedence discovery similarly needs to know the project name, project location, and standardization to use and the dimension to tie the attributes to in the precedence hierarchy. This must be a pipeline dimension to avoid process errors. The precedence discovery looks through the data lenses in the same manner as the dimension discovery to identify the attributes and build a precedence rule assigning the sub-dimension to the parent.

The loader process prepares the pipeline for the actual baseline run. The pipeline adapter and the parser DSA will then work in conjunction to provide the values for those attributes into the pipeline. These two pieces are incorporated and run as a part of the Endeca baseline process. The baseline process is started through your normal mechanism. The pipeline is typically modified to add the `PdqAdapter` after a cache, and the data is then forked off into two routes. A record manipulator strips all the unused properties from the Oracle side of the stream. The `PdqAdapter` java manipulator is then called, and the standardized and attributed data is joined back into the main stream.

The `PdqAdapter` calls out to an external java process that manages the flow of information to the Oracle DataLens Server, and adds the return data back into the Endeca pipeline. The data in the pipeline passes through the adapter, and is sent to the Oracle DataLens Server as configured. On the Oracle DataLens Server, the data is processed by the parser DSA. The data is standardized and cleansed according to the rules defined in the data lenses. At the end of the DSA, there must be a single text output returned; there can be additional outputs, as long as the **Do NOT return results to caller** is selected on the Output Information tab of the output step. The `PdqAdapter` then receives the results from the Oracle DataLens Server, and returns them as properties into the pipeline. These properties are turned into dimensions by the `PropMapper` (the normal Endeca methodology).

## Supported Versions

The following versions are supported with this release:

Oracle Enterprise Data Quality for Product Data, Release 11g (11.1.1.6.1) and later

Oracle Endeca IAP, Version 6.0

# 2

# Installing the Endeca Connector

This chapter describes how to extract and install the Endeca Connector jar files on both Linux and Windows platforms.

The Endeca Connector installation directory, `opdq-connector-endeca`, is contained in the `opdq-connectors-11-1-1-6.zip` file that is installed as part of the EDQP 11g R1 (11.1.1.6.1) release. For installation instructions, see *Oracle Enterprise Data Quality for Product Data* Oracle DataLens Server Administration Guide.

The EDQP home directory contains all the components necessary to the product, including the Endeca Connector installation files. The default installation directory for EDQP is:

On Linux and UNIX: `/opt/Oracle/Middleware/edqp_template1`

On Windows: `C:\Oracle\Middleware\edqp_template1`

This directory path is referenced as the *`EDQP_HOME`* directory in this document.

The Endeca Connector is installed using the following steps:

1. Go to the *`EDQP_HOME`* directory.
2. Extract the `opdq-connectors-11-1-1-6.zip` file to extract the various installation zip files it contains and the `opdq-connector-endeca` directory:

```
\---opdq-connector-endeca
    |   opdq-connector-endeca.jar
    |
    +---exampleProject
    |       connector_endeca_deleter.pmap
    |       connector_endeca_discovery.pmap
    |       dlsapp.zip
    |       dlsapp_parser.pmap
    |
    \---lib
            jdom-1.0.jar
            opdq-api1.jar
            opdq-core.jar
```

   The `opdq-connector-endeca` directory contains the Endeca Connector jar files for use by the Forge process and an example Endeca Connector implementation (see "Setting Up an Example Endeca Connector Project" on page D-1.)

3. On your Endeca server, you must create a directory in which you will place the Endeca Connector jar files. For example, create a directory called `edqp` in the main Endeca folder, and then copy the `opdq-connector-endeca` directory into the `edqp` directory. This creates the following.

| Name | Description |
|---|---|
| `lib` directory | Contains the EDQP API library files. |

| Name | Description |
|---|---|
| `exampleProject` directory | Contains the example Endeca Connector implementation files. |
| `opdq-connector-endeca.jar` | Contains the libraries for the PDQ-Endeca Connector DSA Transform Add-In and the PDQ-Endeca Connector Adapter. |

4. In your WebLogic Server Administration Console, restart the **dls_domain** to reload the `xml` configuration files.

# 3

# Setting Up and Configuring

This chapter describes how to set up and configure the Forge process to use the Endeca Connector jar files.

## Setting Up the Endeca Connector Adapter in the Endeca Development Studio

This section explains the following:

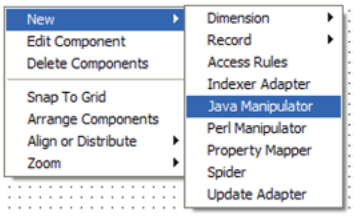
- Adding the Endeca Connector Adapter
- Shared Parameters with the Discovery DSA

### Adding the Endeca Connector Adapter

The Endeca Connector Adapter is the part of the Endeca pipeline that dynamically calls EDQP to retrieve the data lens attributes for each line of input data. These attributes are then mapped to dimensions in Endeca. The Endeca Connector Adapter must be added to your pre-existing pipeline process.

Add the Endeca Connector Adapter to the pipeline:

1. Right-click on the "Pipeline Diagram" and select the **Java Manipulator**.




2. Name the adapter `PdqAdapter`.

   This name is important because it provides validation with the Endeca Connector attribute discovery to verify servers, DSAs, and standardizations.

3. Add the Class as:

   ```
   oracle.pdq.dlfoundry.adapter.PdqAdapter
   ```

4. Add the pathnames to the Endeca Connector (`PdqAdapter`) libraries as in the following example.

   ```
   /Endeca/edqp/lib/opdq-api.jar
   /Endeca/edqp/lib/jdom-1.0.jar
   /Endeca/edqp/lib/opdq-core.jar
   ```

```
/Endeca/edqp/opdq-connector-endeca.jar
```

The pathnames must match the installed location you created in "Installing the Endeca Connector" on page 2-1. These directories may be different than the example if you installed the Endeca Connector files into a different directory.

---

**Note:** In a separate server installation where the Oracle DataLens Server is installed on a separate machine than the Endeca IAP server, the files in the `$MW_HOME\edqp_template1\opdq-connector-endeca` directory must be moved to the Endeca IAP server machine.

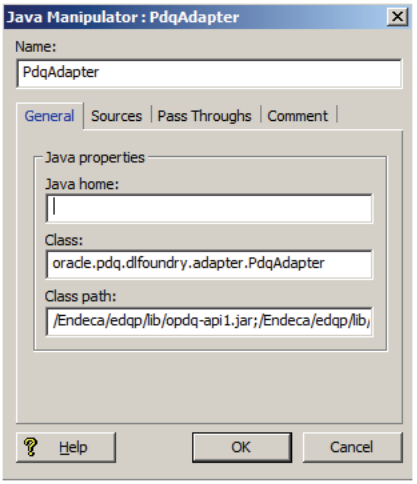
---

---

**Note:** A Windows installation requires the use of a semi-colon ";" as the separator between the paths for each jar file; however, a Linux installation requires the use of a colon ":" as the separator. For example, the Windows path may be,

```
/Endeca/edqp/lib/opdq-api1.jar;/Endeca/edqp/lib/jdom-1.0.jar
;/Endeca/edqp/lib/opdq-core.jar;/Endeca/edqp/opdq-connector-
endeca.jar
```

As in the following example:




---

5. Click the Pass Throughs tab and add the following:
   - **PDQ_SERVER_1** - This is the name or IP address of the Oracle DataLens Production server and the port (*server*:2229).
   - **PDQ_SERVER_2** - This is an optional server for use in high availability and load balancing. For more information, see Appendix C, "Endeca Connector Robustness."
   - **PDQ_SERVER_3** - This is an optional server for use in high availability and load balancing (note that there is *no* limit to the number of `PDQ_SERVER_*` entries)
   - **DSA_MAP** - This is the top-level DSA to call on the Oracle DataLens Server.
   - **DSA_OUTPUT_STEP** - This is an optional step name if the DSA has multiple outputs.
   - **LOCALE** - This is the input locale of the data.

- **BATCH_SIZE** - The number of records to process in a single chunk.
- **PROPERTY_ID** - The name of the ID field in the input data.
- **PROPERTY_ROUTE_INFO** - The "hint" used to efficiently route the data.
- **PROPERTY_DESC1** - The name of the first description field in the input data.
- **PROPERTY_DESC2** - The name of the second description field in the input data.
- **PROPERTY_ALT1** - 1st alternate data field (`mfgName`).
- **PROPERTY_ALT2** - 2nd alternate data field (`mfgPartNo`).
- **PROPERTY_ALT3** - 3rd alternate data field (user-defined).
- **RETURN_VAL1** - 1st return value from the DSA (after the ID).
- **RETURN_VAL2** - 2nd return value from the DSA.
- **RETURN_VAL3** - 3rd return value from the DSA.
- **REPLACE_UNDERSCORES_ONLY** - A value of true will *not* proper case the attribute names, false will proper case the attributes. In either case, the underscores will be replaced with spaces.

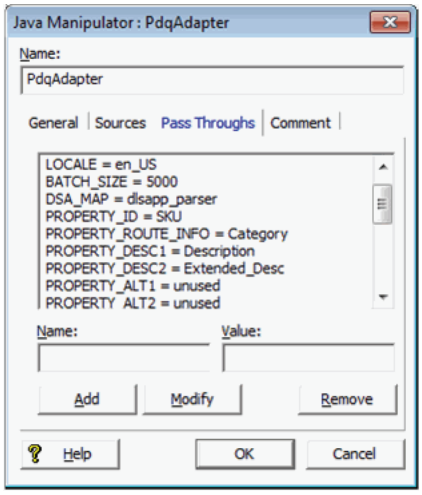
  **Note:** This parameter is used by both the Endeca Connector Discovery processes and the `PdqAdapter`.

- **USE__PREFIX** - A value of true will put a "PDQ" prefix on all the attributes discovered; false will not.

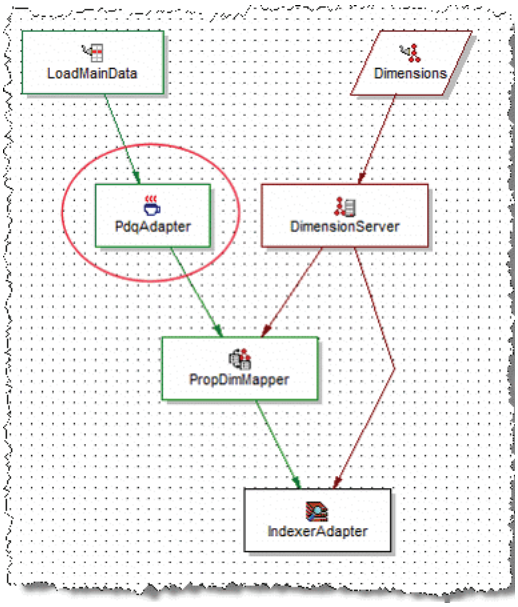
  **Note:** This parameter is used by both the Endeca Connector Discovery processes and the `PdqAdapter`.

- **DEBUG** - This toggles (`true/false`) debug tracing to a log file on or off. This should only be turned on when debugging, because it will slow down the Endeca Connector Adapter. The following log file is created in the Endeca project directory:

  `Edf.Pipeline.RecordPipeline.JavaManipulator.PdqAdapter.log`



6. Click **OK**.
7. Insert the `PdqAdapter` into the pipeline flow between the load data nodes and the Property Mapper.
8. Insert the `PdqAdapter` directly below the **LoadData** step in the Endeca Pipeline flow to ensure that the Endeca Connector Adapter only calls the Oracle DataLens Server a single time for each line of data to be processed as in the following:



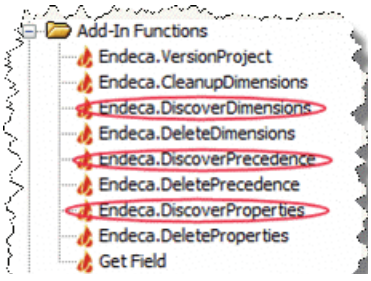

9. Save your project.

## Shared Parameters with the Discovery DSA

Several of the Endeca Connector Discovery Add-In Functions share some of the pass through parameters from the `PdqAdapter`. The Add-In Functions are:

- `Endeca.DiscoverDimensions`
- `Endeca.DiscoverProperies`
- `Endeca.DiscoverPrecedence`

This simplifies the configuration and keeps the `PdqAdapterPdqAdapter` in sync with the DSA.




The following `PdqAdapter` pass through parameters are those that are shared:

- **PDQ_SERVER_1**
- **PDQ_SERVER_2** (optional)

- **PDQ_SERVER_3** (optional)
- **DSA_MAP**
- **REPLACE_UNDERSCORES_ONLY**
- **USE_PDQ_PREFIX**

# Configuring the Endeca Connector Attribute Discovery with Application Studio

The Endeca Connector looks at the DSA that is being run as part of the Forge processing, and then determines exactly which data lenses are being called and exactly which standardizations are being run and exactly which attributes are being extracted. This unique list of attributes is then used to update the internal Endeca Dimensions and Properties by updating the configuration files used in the Endeca Project.

This section explains the following:



These sections creates the DSA that is necessary to the Forge processing and are progressive.

## Configuring the Dimension Discovery

1. Start your Oracle DataLens Server.
2. Open one of the supported Web browsers for your environment. See the Oracle Enterprise Data Quality for Product Data Certification Matrix at

   `http://www.oracle.com/technetwork/middleware/ias/downloads/fusion-certification-100350.html`

   Locate **Oracle Enterprise Data Quality** in the Product Area column and then click the **System Requirements and Supported Platforms for Oracle Enterprise Data Quality for Product Data 11gR1 (11.1.1.x) Certification Matrix (xls)** link.
3. Enter the following URL:

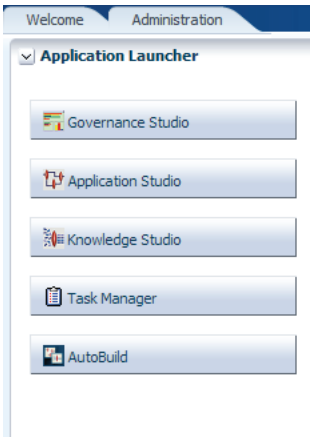
   `http://`*`hostname:port`*`/datalens`

   where *hostname* is the DNS name or IP address of the Administration Server and *port* is the listen port on which the Administration Server is listening for requests (port 2229 by default).

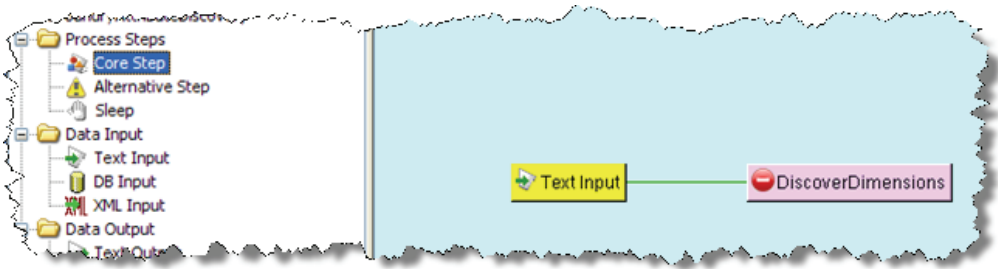
   If you configured the Administration Server to use Secure Socket Layer (SSL) you must add s after http as follows:

   `https://`*`hostname:port`*`/datalens`
4. When the login page appears, enter a user name and the password. Typically, this is the user name and password you specified during the installation process.

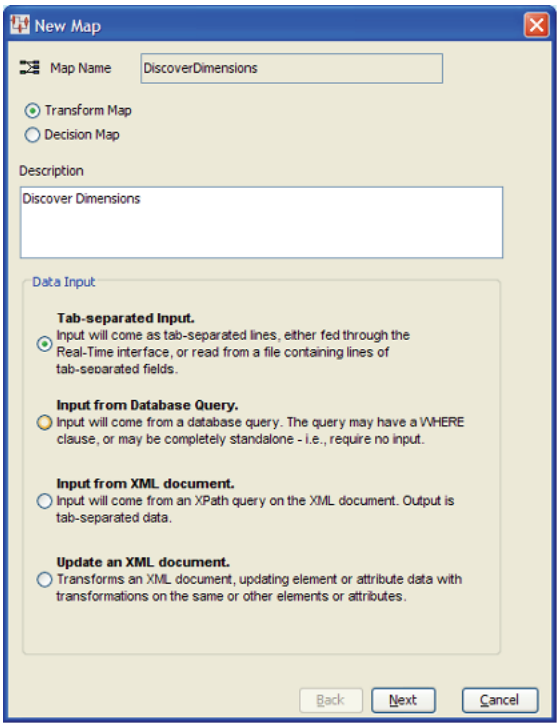
The Oracle DataLens Server Web pages are displayed and default to the **Welcome** tab.



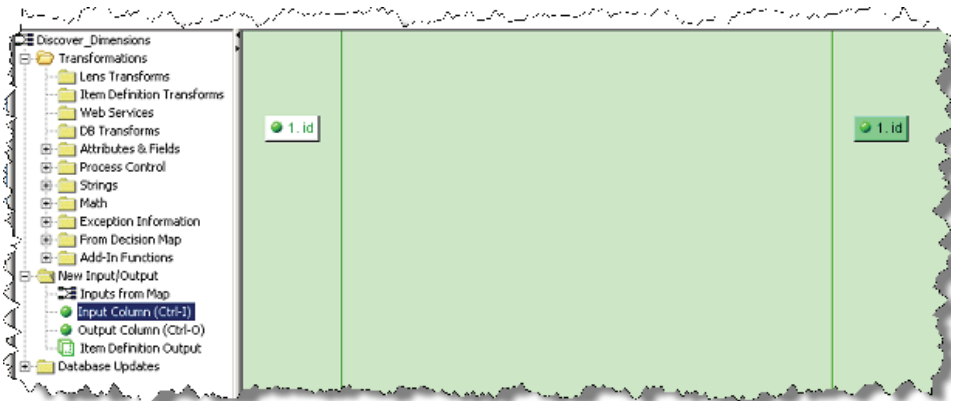

5. Click the Application Studio button.
6. Create a new project, add an input step and a core step. In this example, the core step is named **DiscoverDimensions.**



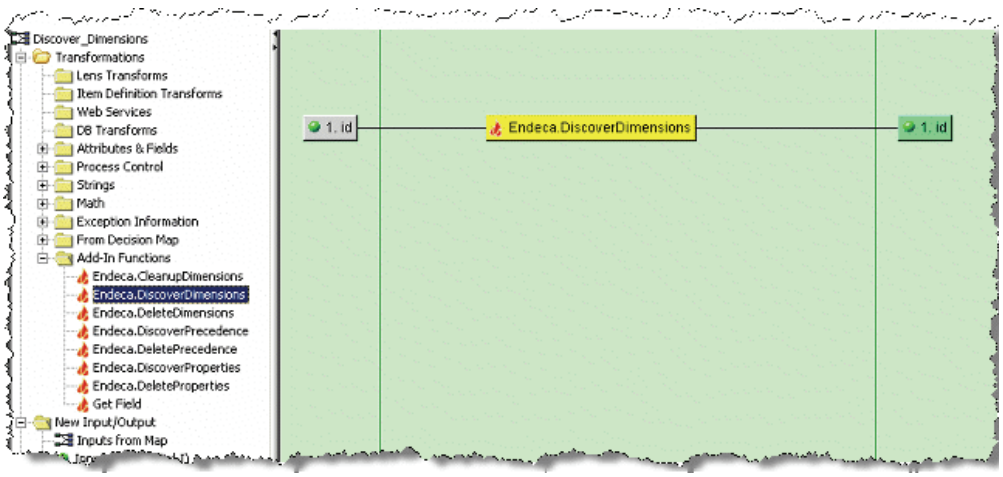

7. Create the new Transform Map by double-clicking on the **DiscoverDimensions** step and select the default, **Tab-separated Input**.




8. Add an input column and an output column as follows:




9. Expand the Add-In Functions, and drag over the **Endeca.DiscoverDimensions**, add it as a transformation to the Transform Map and connect the Transform Map steps as follows:




**10.** Configure the Endeca.DiscoverDimensions Add-In adapter by double clicking on the **Endeca.DiscoverDimensions** step and using the **Parameters** tab.

Any of the values in the **Value** column may be edited except the `StartDimRange` and `EndDimRange` values.

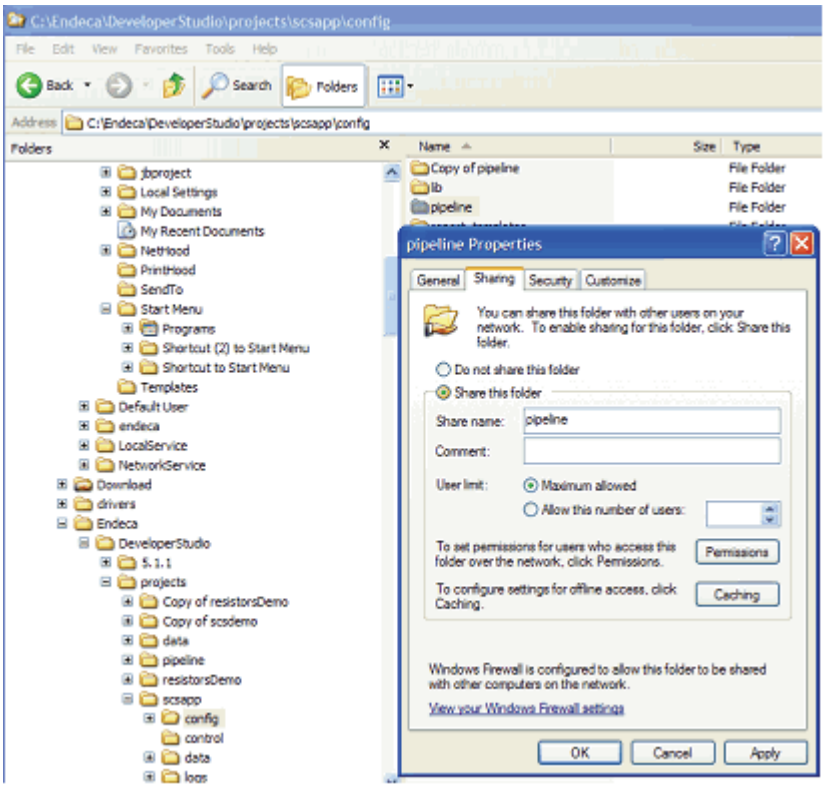
### Endeca Project Parameters

Main | Parameters

| Name | Value | Description |
|---|---|---|
| Endeca.ProjectName | dlsapp | Endeca Project Name |
| Endeca.ProjectLocation | C:\Endeca\projects\dlsapp\c... | Endeca Project Location |
| Endeca.StartPdqDimRange | 100000 | The STARTING range value to use for PDQ-Generated Endeca Dimension Id values in a Dev Studio Project |
| Endeca.EndPdqDimRange | 200000 | The ENDING range value to use for PDQ-Generated Endeca Dimension Id values in a Dev Studio Project |
| Endeca.ExternalDimensionsFile | externaldimensions.xml | Externally created Dimensions that have not been loaded into the Endeca Project (leave blank to omit) |
| Endeca.PipelineEpxFile | pipeline.epx | Endeca Project pipeline configuration file |
| Endeca.DimensionsXmlFile | dimensions.xml | Endeca Project dimensions configuration and definition file |
| ---------- | ---------- | ---------- |

For a DSA that is running on a separate server machine from the Endeca Server, an UNC pathname must be used as in the preceding example, or a mounted file system if in a Linux environment.

On the Endeca Server, share the project pipeline directory for access by the Endeca Connector as follows:




Be sure to set the permissions so that the Endeca Connector process can access this directory and has access rights to the files as well. In other words, the path to the project from the Endeca output adapters is:

```
\\endeca51Server\pipeline
```

## DSA Project Parameters

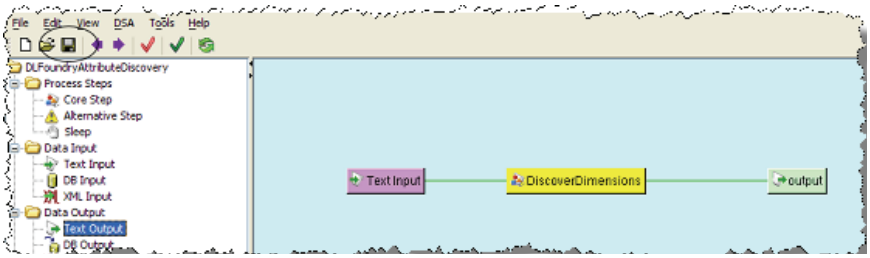
| | | |
|---|---|---|
| PDQ.Standardization | Match_Attributes | The PDQ Standardization used by the DataLenses for discovering Endeca Dimensions |
| PDQ.Tracing | false | (true, false) Trace the attributes used by PDQ to discover the Dimensions down to the DataLens, Frame, and ItemName level. |
| PDQ.VersionEndecaFiles | false | (true, false) Version the original Endeca configuraion files with a time-stamped version when the PDQ-Created Dimensions are generated |
| PDQ.DeleteUnusedDimensions | false | (true, false) Delete any Endeca dimensions created by PDQ that are no longer being used in any DataLenses. |
| ---------- | ---------- | ---------- |

### Endeca Dimension Creation Parameters

| | | |
|---|---|---|
| Search.SearchDim | true | (true, false) Specifies whether or not record search should be enabled for this dimension. Record search finds all records in an Endeca application that have a dimension whose value matches a term the user provides. Checking 'Enable record search' makes the following additional search options available. |
| Search.ClassifyDim | false | (true, false) When true, allows record search to consider ancestor dimension values when matching a record search query. This setting is only enabled when 'Enable record search' is true. |
| Search.DisplayDim | true | (true) Toggles the display of Dimensions. This option should always be set to true or the Web App will get no dimensions displayed. |
| Search.HierarchyForRecord | false | (true, false) When true, allows record search to consider ancestor dimension values when matching a record search query. This setting is only enabled when "Enable record search" is true. |
| Search.HierarchyForDimension | false | (true, false) When true, allows dimension search to consider ancestor dimension values when matching a dimension search query. This setting is independent pf "Enable record search". |
| General.ShowWithRecordList | true | (true, false) When true, enables this dimension to appear in the record list display. Any records that are tagged with a value from this dimension will have the value shown as part of their entry in the record list. |
| General.ShowWithRecord | false | (true, false) When true, allows this dimension to appear on the record page. Any records that are tagged with a value from this dimension will have that value shown as part of their entry on the record page. |
| Advanced.ComputeRefinementStatistics | true | (true, false) Dimension statistics count the number of records, for a given result set, that are tagged with each of a dimension's dimension values. This gives the end-user an indication of the number of records that will be returned. |
| Advanced.CollapsableDimensionThreshold | 10 | Allows you to set your application to collapse a deep hierarchy to make it shallower when available data is small. The collapsible dimension threshold determines how many refinement values must exist for a set of query results in order for dimension value collapsing to occur. |
| Advanced.MultiSelect | OFF | (OFF, OR, AND) Allows the end user to select more than one Dimension value from a Dimension. |
| DynamicRanking.EnableDynamicRanking | true | (true, false) When true, indicates that the list of refinement dimension values returned for a query should be pruned to those values that occur most frequently in the requested navigation state; that is, the refinement dimension values that are most |
| DynamicRanking.MaxDimValueToReturn | 10 | Sets the number of most popular dimension values to return. |
| DynamicRanking.SortDimensionValues | ALPHA | (ALPHA, NUMERIC) Establishes the sort method used for the most popular dimension values: 'Alphabetically' uses whatever order you've selected for the 'Refinements sort order' setting. 'Dynamically' orders the most popular refinement values according to their frequency of appearance within a data set. Dimension values that occur more frequently are returned before those that occur less frequently. |
| DynamicRanking.GenerateMore | true | When this option is true, if the actual number of refinement options exceeds the number set in 'Maximum dimension values to return', then an additional option called More is returned for that dimension. If the user selects the More option, then the Navigation Engine will return all of the refinement options for that dimension. If 'Generate More' is not set, only the number of dimension values defined in 'Maximum dimension values to return' is displayed. |
| DynamicRanking.RefinementSortOrder | DYNAMIC_RANK | (DYNAMIC_RANK, DEFAULT) Specifies the sort type for any refinement dimension values that are returned for this dimension: Alpha, Integer, or Floating point. Default dimension value ranking is used with dimensions that are auto-generated. In an auto-generated dimension, you don't have direct access to and, hence, can't manually rank, the dimension values. Instead, you must set a default rank order. |
| DynamicRanking.DynamicRankingType | FREQUENCY | (FREQUENCY) |

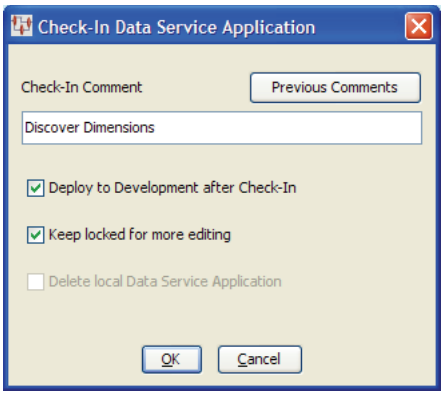
Save and close the Transform Map.

Add an output step to the DSA.




Save the DSA.

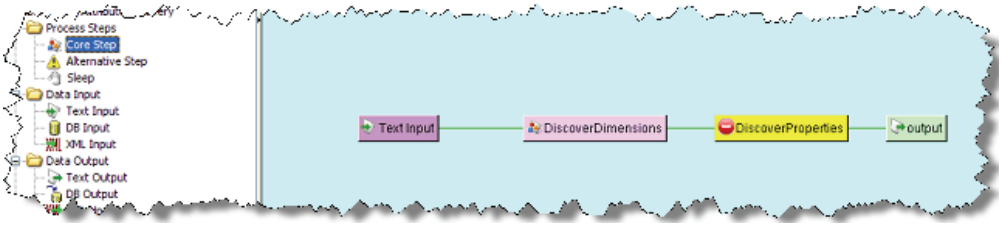
Check-in the DSA to the Oracle DataLens Server to make it available for processing.




## Configuring the Property Discovery

Create a new core step in your DSA called **DiscoverProperties**.

Add a new **Endeca.DiscoverProperties** Transform Add-In to the Transform Map by dragging the Add-In into our Transform Map and configuring, as previously described.




### Endeca Property Creation Parameters

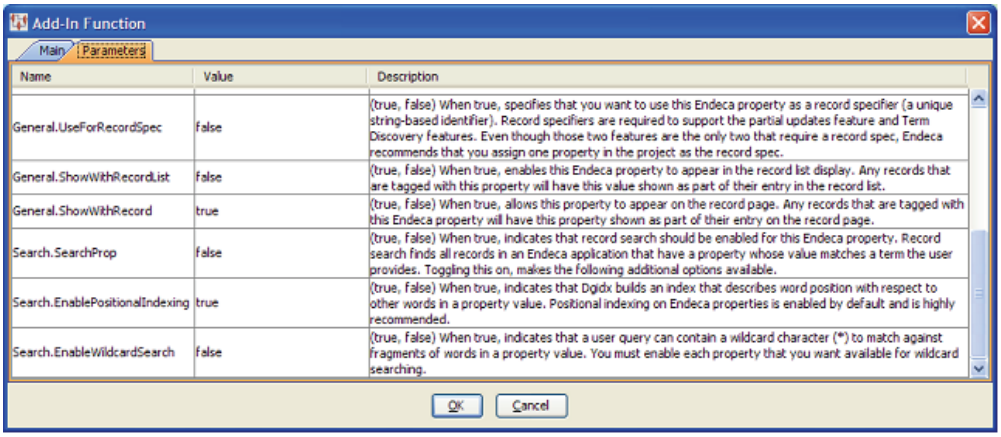
Now configure the **Endeca.DiscoverProperties** Transform Add-In using the **Parameters** tab.

---

**Note:** The `Endeca.* Parameters` are the same as for `Endeca.DiscoverDimensions`.

---

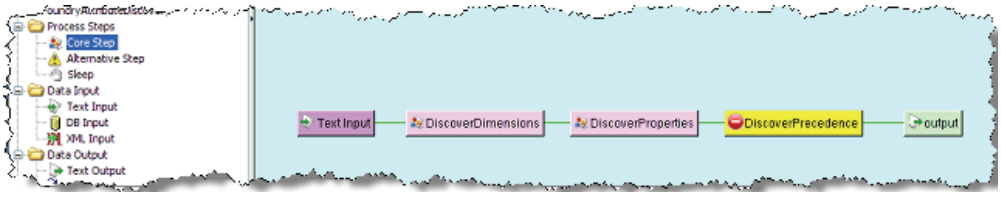
Any of the values in the **Value** column may be edited.



## Configuring Precedence Rules for the Entire Project

Create a new core step in your DSA called **DiscoverPrecedence**.

Add a new **Endeca.DiscoverPrecedence Transform** Add-In to the Transform Map by dragging the Add-In into your Transform Map and configuring, as previously described.




### Endeca Precedence Rule Creation Parameters

Now configure the **Endeca.DiscoverPrecedence** Transform Add-In using the **Parameters** tab.

Any of the values in the Value column may be edited.

| | | |
|---|---|---|
| SourceDimension | Product Category | The dimension that serves as a trigger for the To (or target) dimension value to be displayed. |
| PrecedenceType | STANDARD | (STANDARD, LEAF) STANDARD means that if the dimension value specified as the trigger or any of its descendents are in the navigation state, then the target is presented (one trigger, one target). LEAF means that querying any leaf dimension value from the trigger dimension will cause the target dimension value to be displayed (many triggers, one target). |

### Determining the Parent (Source Dimension)

The first step is to determine which Dimension you would like as the source or Parent Dimension. This is the Dimension that will need to be selected to enable the activation of the other -generated Dimension in your Endeca-powered web site.

First, run the PDQ-Endeca Connector program to create the Dimensions.

Second, review the output from the Endeca Connector program and select the Dimension that you would like to use for the source. For example “Product Category” as in the following example:

> **Note:** You could look at the dimensions in the Endeca Development Studio to obtain a source dimension that may not have been created.

```
PDQ-Endeca Connector Version 11.1.1.6.0, Build 20120804 Copyright (c) 2008, 2012,
 Oracle and/or its affiliates. All rights reserved.
Running on DataLens Administration server endeca01:2229
Extracted 197 distinct attributes from the PDQ-Endeca Connector.
Added 197 new Dimensions.
Accessory Component Quantity with Id of: 100001
…
Product Category with Id of: 100134
…
Wood Species with Id of: 100197
Wood Species with Id of: 100197
```

Save the DSA.

Check in the DSA to the Oracle DataLens Server to make it available for processing.

### Creating Separate Precedence Rules for Each Data Lens

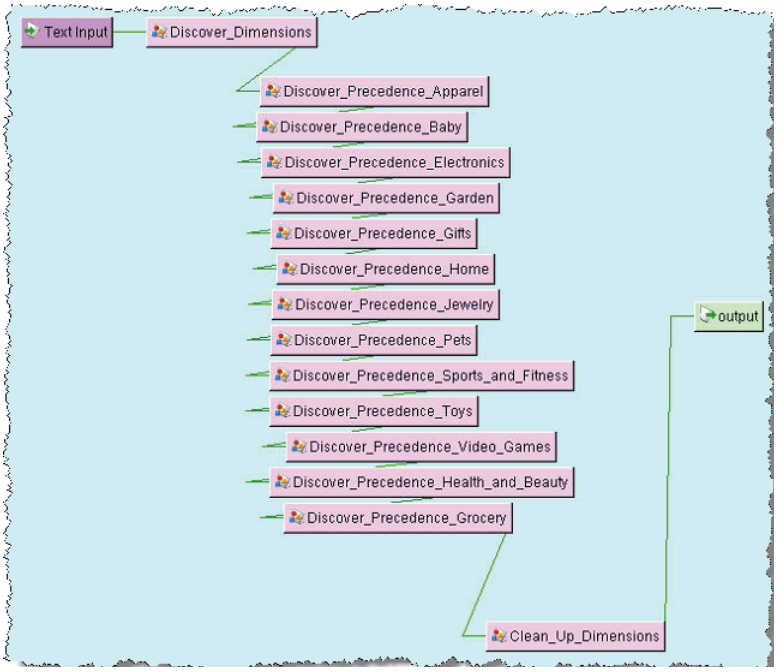
This operates the same as creating precedence rules as previously described.

The difference is that the `PDQ.dataLens` value will be set to the name of a valid data lens used in the main DSA. You would then need to create a separate step in the DSA for each Precedence mapping. Therefore, if you have 12 data lenses being used in the DSA, you would have 12 DiscoverPrecedence Transform Map steps to process all the rules.

| | | |
|---|---|---|
| PDQ.DataLens | Electronic_Components | Attributes from this DataLens are used to define which Endeca Dimensions get these precedence rules. Leave this field empty to create precedence rules for ALL PDQ-generated Dimensions. This should be a DataLens used by the DSA map used in Endeca.DiscoverDimensions. |
| PDQ.Standardization | Match_Attributes | The PDQ DataLens Standardization used to define which Endeca Dimensions get these precedence rules (only if the PDQ.DataLens is defined) This should be the Standardization used by in Endeca.DiscoverDimensions. |
| PDQ.GenerateRulesForAllAttributes | true | (true, false) If true, then precedence rules are also created for dimensions not generated by the PDQ DL Foundry if they are in the list of attributes for the specific DataLens. This option only applies when the PDQ.DataLens parameter is defined. Note that this will never create duplicate rules if they have been manually created. |
| PDQ.Tracing | false | (true, false) Trace the attributes used by PDQ to discover the Properties down to the DataLens, Frame, and ItemName level. |
| PDQ.VersionEndecaFiles | false | (true, false) Version the original Endeca configuraion files with a time-stamped version when the PDQ-Created Precedence rules are generated |
| PDQ.DeleteUnusedPrecedenceRules | true | (true, false) Delete any Endeca Precedence Rules created by PDQ that are no longer being used in any DataLenses. |
| ---------- | ---------- | ---------- |
| SourceDimension | Product Category | The dimension that serves as a trigger for the To (or target) dimension value to be displayed. |
| PrecedenceType | STANDARD | (STANDARD, LEAF) STANDARD means that if the dimension value specified as the trigger or any of its descendents are in the navigation state, then the target is presented (one trigger, one target). LEAF means that querying any leaf dimension value from the trigger dimension will cause the target dimension value to be displayed (many triggers, one target). |

In addition, another parameter is used when creating separate precedence rules for each individual data lens. This is the `PDQ.GenerateRulesForAllAttributes`. This will toggle on the creation of precedence rules for non-PDQ generated Dimensions if they are found in the particular data lens.

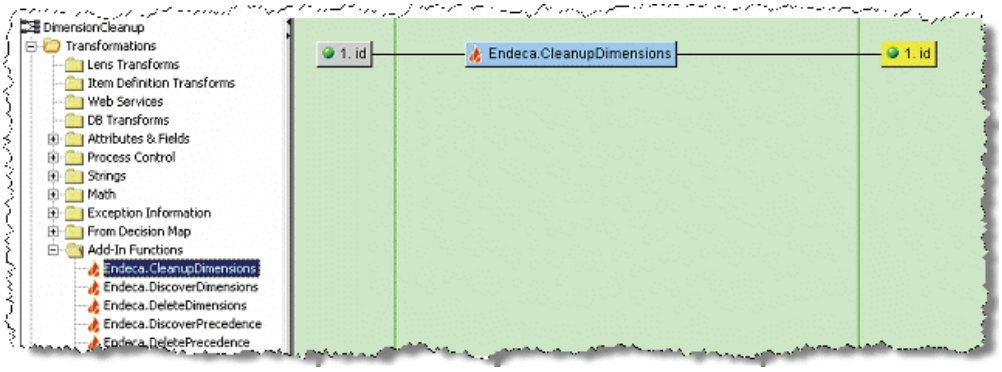
Following is an example DSA with precedence rule discovery being done for each individual data lens:



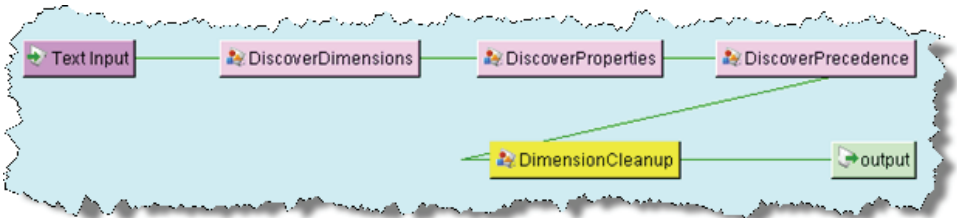
**Note:** If you toggle on `.VersionEndecaFiles`, that this should only be done for the first of the multiple precedence rule steps to avoid creating unnecessary backup files.

## Adding a Cleanup Step

This step will free up the dimension information from the DSA and the dimension information from the Endeca Project, including Dimensions from the Endeca External Dimensions file. This data is cached in the Oracle DataLens Server for speed in processing the multiple steps in the Endeca Connector Discovery jobs. The **Endeca.CleanupDimensions** step removes the data structures used by the Endeca Connector so that the dimensions and properties used in this discovery run are not used in subsequent discovery runs that may have added or deleted attributes in data lenses. This also frees the cached data so there is more memory available the server to use for processing.



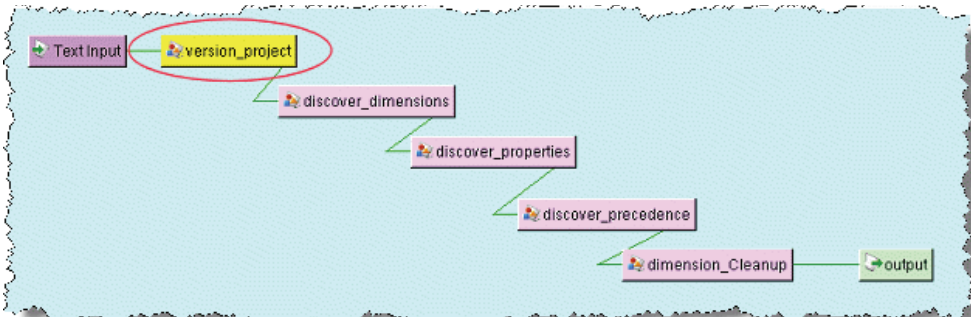

Now, the top-level DSA should look like the following:



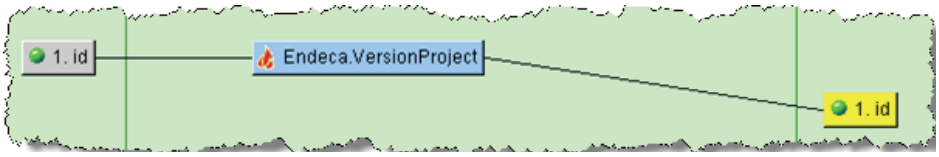

## Adding a Project Versioning Step

This step versions all of the Endeca project files that are updated by the Endeca Connector. This is so that the project can be easily set back to the point where the project was at prior to running the Endeca Connector discovery DSA.

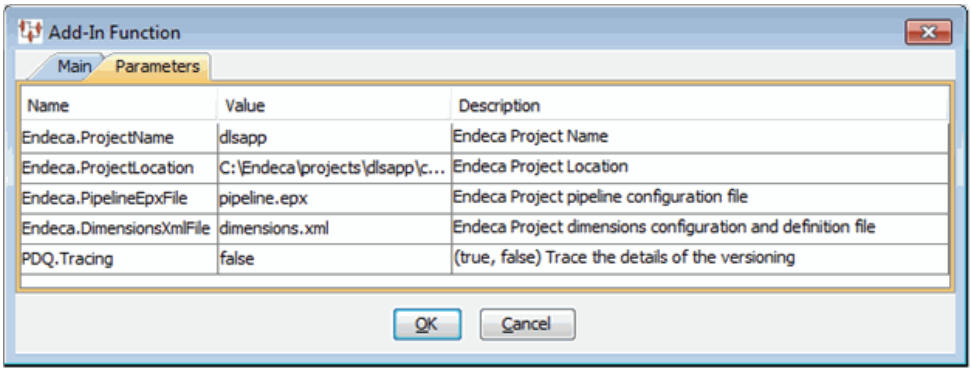
The following example shows a typical DSA Endeca Connector discovery job with the **Endeca.VersionProject** step added.




Following is the **Endeca.VersionProject** XFM map that is associated with the Version Project DSA Step.




There are only three parameters that are needed for the **Endeca.VersionProject** Transform Map Add-In step.

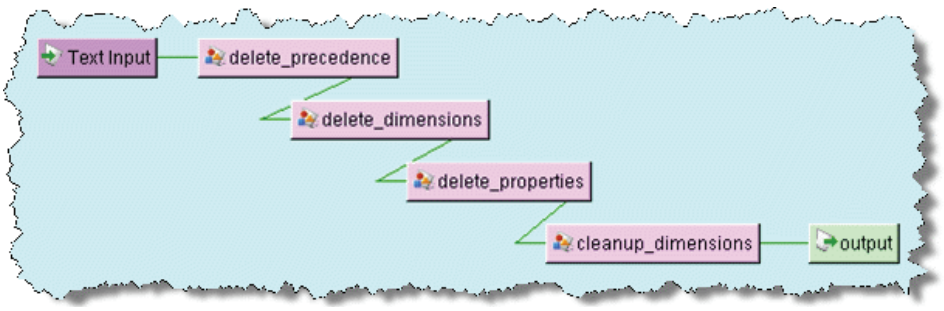



After running this step, you will see the following message in the Oracle DataLens Server Log file showing the file suffix that was appended to a copy of the project files.

```
PDQ-Endeca Connector  Versioned the Endeca project files with a suffix of
20121005104908
```

## Removing Generated Values

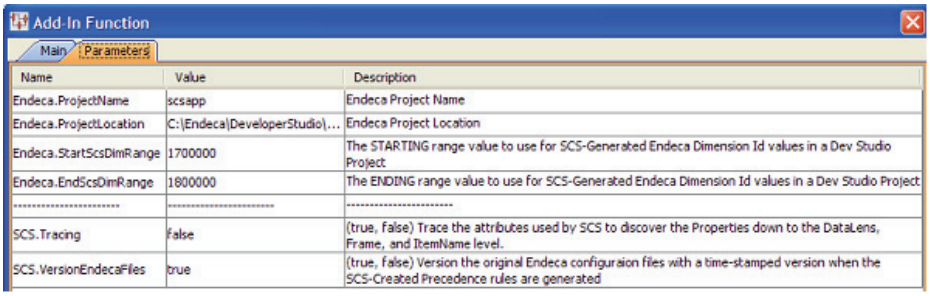
Any of the Dimensions, Properties, or Precedence Rules can be deleted with a Deletion DSA as follows:



**Note:** The Precedence Rules should be deleted *before* the Dimensions. This is because the Precedence Rules are built on the Dimensions that were created by the Endeca Connector.

The Properties have no dependencies and can be deleted at any step in the DSA.

The parameters for the delete functions still need the Endeca project information and just the flags for tracing and versioning. The start and end ranges cannot be changed.




## Changing the Dimension Id Range Values Used by the Endeca Connector

The range values are set in the DSA as a hard-coded range from 100,000 to 200,000. These values should never be changed unless they conflict with values already being used by Endeca. These values cannot be change in the Endeca Connector DSA Add-Ins (the values in the DSA are read-only).

The two values that set the range are as follows:

- `Endeca.StartPdqDimRange`
- `Endeca.EndPdqDimRange`

| | | |
|---|---|---|
| Endeca.StartScsDimRange | 1700000 | The STARTING range value to use for SCS-Generated Endeca Dimension Id values in a Dev Studio Project |
| Endeca.EndScsDimRange | 1800000 | The ENDING range value to use for SCS-Generated Endeca Dimension Id values in a Dev Studio Project |

**Note:** In the previous example, the values have been changed to 1,700,000 and 1,800,000.

To change these values do the following:

1. Delete all the -generated precedence rules and -generated dimensions from the Endeca project using a delete DSA.
2. Stop the Oracle DataLens Server.
3. Edit the `AddInTransformParameters.xml` configuration file.

   This file resides in `$EDDQ_HOME\config` where `$EDQP_HOME` is the directory in which you installed the EDQP product.
4. Change the default values for the **Endeca.StartPdqDimRange** and the **Endeca.EndPdqDimRange** in the four places each appears.
5. Start the Oracle DataLens Server
6. Recreate the DSAs used by the Endeca Connector attribute discovery and attribute delete Transform Add-Ins.
7. Save and check-in each new DSA.

8. Run the Discover attribute DSA on the Endeca project.

# 4

# Running the Endeca Connector

This chapter explains how you run the Endeca Connector.

## Running the Endeca Connector Interactively

The Endeca Connector can be run using the DSAs that you created in Chapter 3, "Setting Up and Configuring". The four easiest methods of running the Endeca Connector to run the DSA from:

- The Oracle DataLens Server Administration web page
- The Governance Studio Application
- Using the Command Line Interface program
- The Services for Excel application

For more information, see the appropriate guide in "Related Documents" on page 0-v.

1. Ensure that your Oracle DataLens Server is running.
2. Open one of the following supported Web browsers for your environment:
   - Internet Explorer 8.0 or later and Internet Explorer 9.0 or later
   - Mozilla Firefox 4.0 or later and Firefox 5.0 or later
   - Google Chrome 12.0 or later
   - Safari 5.0 or later
3. Enter the following URL:

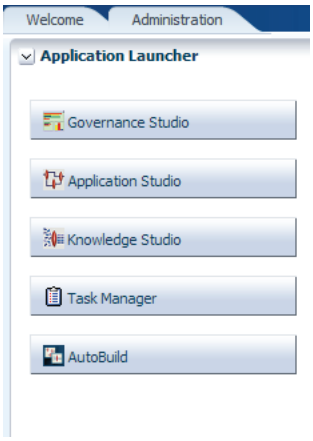
   `http://`*`hostname:port`*`/datalens`

   where *hostname* is the DNS name or IP address of the Administration Server and *port* is the listen port on which the Administration Server is listening for requests (port 2229 by default).

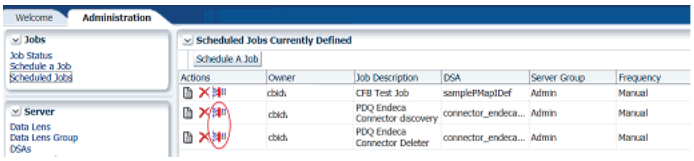
   If you configured the Administration Server to use Secure Socket Layer (SSL) you must add s after http as follows:

   `https://`*`hostname:port`*`/datalens`
4. When the login page appears, enter a user name and the password. Typically, this is the user name and password you specified during the installation process.

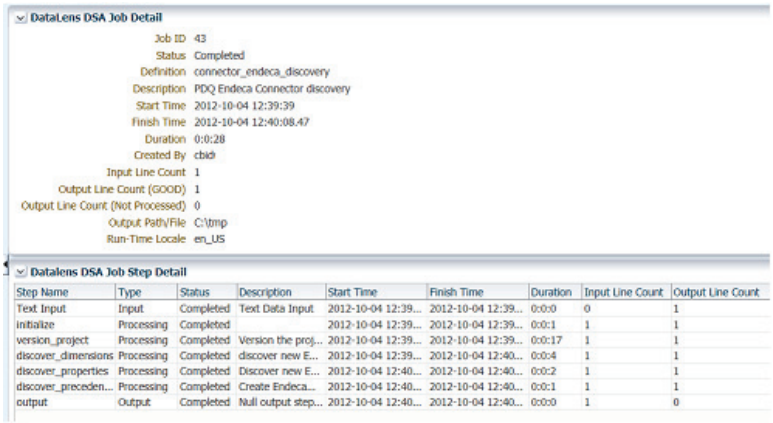
   The Oracle DataLens Server Web pages are displayed and default to the **Welcome** tab.




5. Select the **Administration** tab.
6. From the **Jobs** panel, select **Scheduled Jobs**, and then click the **Run** button adjacent to the DSA job you want to run.




Following is the output from the Endeca Connector Interactive Job.




# Running the Endeca Baseline Update

At this point, the Endeca Project configuration files have been updated. This data is now ready for use in the Endeca Developer Studio or when running the Endeca Baseline update.

Execute the script used to run the Endeca Baseline Update.

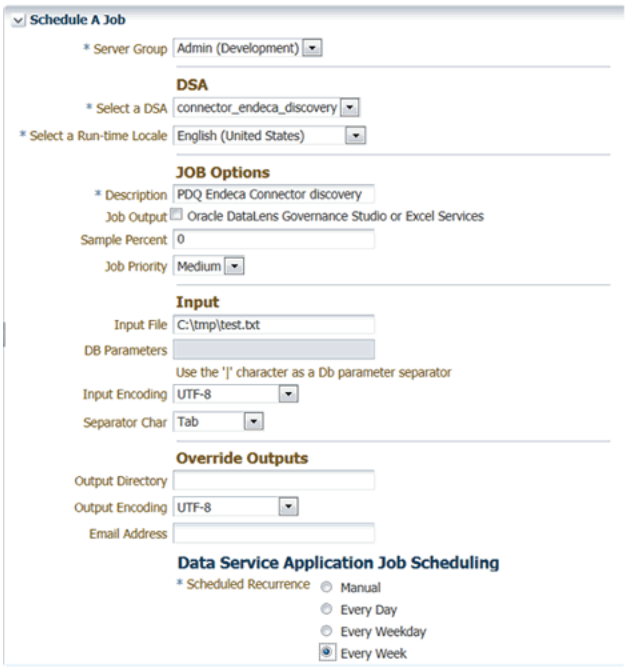
In the Endeca IAP this is done with the Deployment Template scripts in the project:

1. Run **`load_baseline_test_data.bat`**.
2. Run **`baseline_update.bat`**.

# Running the Endeca Connector in Batch Mode

Scheduling Endeca Connector jobs is done with the DSA Job scheduler. This can be done from the Administration Web pages or the AMS application.

Following is an example of a scheduled job from the Administration web page to run a Endeca Connector Discovery weekly:



# 5

# Logging and Tracing Data

This chapter explains how information is logged and how you can trace data.

## Logging

The Endeca Connector writes the basic processing information to the Oracle DataLens Server log file as it is processing data.

1. Ensure that your Oracle DataLens Server is running.
2. Open one of the following supported Web browsers for your environment:
   - Internet Explorer 8.0 or later and Internet Explorer 9.0 or later
   - Mozilla Firefox 4.0 or later and Firefox 5.0 or later
   - Google Chrome 12.0 or later
   - Safari 5.0 or later
3. Enter the following URL:

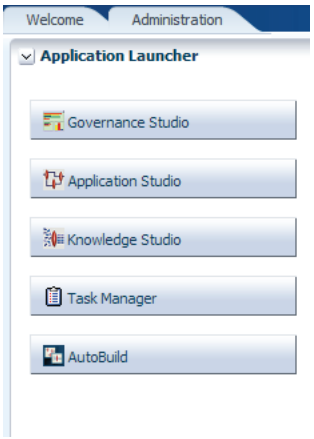
   `http://`*`hostname:port`*`/datalens`

   where *hostname* is the DNS name or IP address of the Administration Server and *port* is the listen port on which the Administration Server is listening for requests (port 2229 by default).

   If you configured the Administration Server to use Secure Socket Layer (SSL) you must add s after http as follows:

   `https://`*`hostname:port`*`/datalens`
4. When the login page appears, enter a user name and the password. Typically, this is the user name and password you specified during the installation process.

   The Oracle DataLens Server Web pages are displayed and default to the **Welcome** tab.




5. Select the **Administration** tab.
6. From the **Admin Dashboard** panel, click the **Admin Log**.

   Following is an example of the logging entries that are output from the Endeca Connector Dimension Discovery to the server log.

```
INFO 20 Oct 2012 15:51:26 [] - Manually Running Scheduled Job Data Service
Application job (4)
WARN 20 Oct 2012 15:51:26 [] -  PDQ-Endeca Connector WARNING: The Parameter
Search.HierarchyForDimension is empty!
INFO 20 Oct 2012 15:51:26 [] -  PDQ-Endeca Connector Version 11.1.1.6.0, Build
20120804 Copyright (c) 2012, 2012, Oracle and/or its affiliates. All rights
reserved. Running on  DataLens Admin server 127.0.0.1:2229
INFO 20 Oct 2012 15:51:26 [] -  PDQ-Endeca Connector - Dimension Discovery
Definining NEW -Generated Dimensions...
INFO 20 Oct 2012 15:51:26 [] -  PDQ-Endeca Connector Extracted 42 distinct
attributes from the Pipeline.
INFO 20 Oct 2012 15:51:26 [] -  PDQ-Endeca Connector Added 42 new Dimensions:
INFO 20 Oct 2012 15:51:26 [] -  PDQ-Endeca Connector  PDQ-Endeca Connector
Accessories with Id of: 100001
INFO 20 Oct 2012 15:51:26 [] -  PDQ-Endeca Connector  PDQ-Endeca Connector
Brand Or Model Name with Id of: 100002
INFO 20 Oct 2012 15:51:26 [] -  PDQ-Endeca Connector  PDQ-Endeca Connector
Candleholder with Id of: 100003
INFO 20 Oct 2012 15:51:26 [] -  PDQ-Endeca Connector  PDQ-Endeca Connector
Clothing with Id of: 100004
INFO 20 Oct 2012 15:51:26 [] -  PDQ-Endeca Connector  PDQ-Endeca Connector
Color with Id of: 100005
INFO 20 Oct 2012 15:51:26 [] -  PDQ-Endeca Connector  PDQ-Endeca Connector
Diameter with Id of: 100006
INFO 20 Oct 2012 15:51:26 [] -  PDQ-Endeca Connector  PDQ-Endeca Connector
Dimensions with Id of: 100007
INFO 20 Oct 2012 15:51:26 [] -  PDQ-Endeca Connector  PDQ-Endeca Connector
Footwear with Id of: 100008
…
INFO 20 Oct 2012 15:51:27 [] -  PDQ-Endeca Connector Completed processing
configuration data in 1.125 seconds
```

   There are similar log entries for the Property Discovery and Precedence Rule Discovery. There are also entries added when anything is deleted by the Endeca Connector.

7. Check for errors or warnings in the log file for problems with the Endeca Connector load process. Notice that in the preceding example, there is a warning.

# Tracing

The section describe how to use tracing in the Endeca Connector DSAs.

## Endeca Connector Attribute Discovery

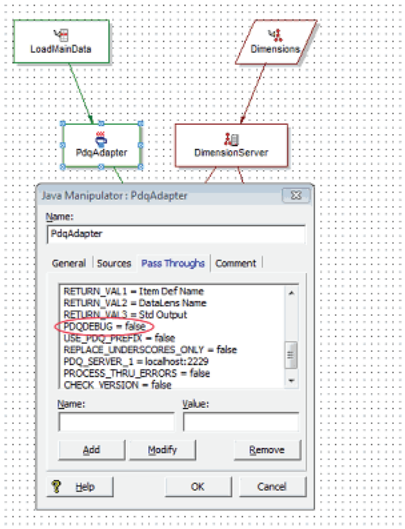
The Endeca Connector Attribute Discovery DSA outputs all the information on the Attributes that were discovered in the Oracle DataLens Server and exactly how these are mapped to the Endeca dimensions and properties. Normally tracing does not need to be turned on, unless you are identifying a problem in the Endeca Connector Processing.

You can turn on tracing by setting the DSA Add-In transformation parameter to 'true' from as follows:

| | | |
|---|---|---|
| PDQ.Tracing | false | (true, false) Trace the attributes used by PDQ to discover the Properties down to the DataLens, Frame, and ItemName level. |

## Endeca Connector Adapter

Set the **PDQDEBUG** parameter to 'true', in the Endeca Project pass throughs, to turn on tracing of the Endeca Connector during the Forge processing. The parameter is set in the Endeca Developer Studio as in the following:




This turns on debug tracing so that processing information is written to a log file. You should only set this parameter to true when you are debugging because it slows down the Endeca Connector Adapter.

The following log file is created in the Endeca project directory with all the Endeca Connector tracing information:

```
Edf.Pipeline.RecordPipeline.JavaManipulator.PdqAdapter.log
```

## Processing Data, Even if Errors Are Encountered

There is an optional pass through parameter called `PROCESS_THRU_ERRORS`.

This option is turned on using the following Endeca Java Manipulator pass through:

```
PROCESS_THRU_ERRORS=true
```

If it is not set, then the Endeca Connector behaves normally and generates an exception that stops the Forge processing if any of the Endeca Connector threads encounter an error.

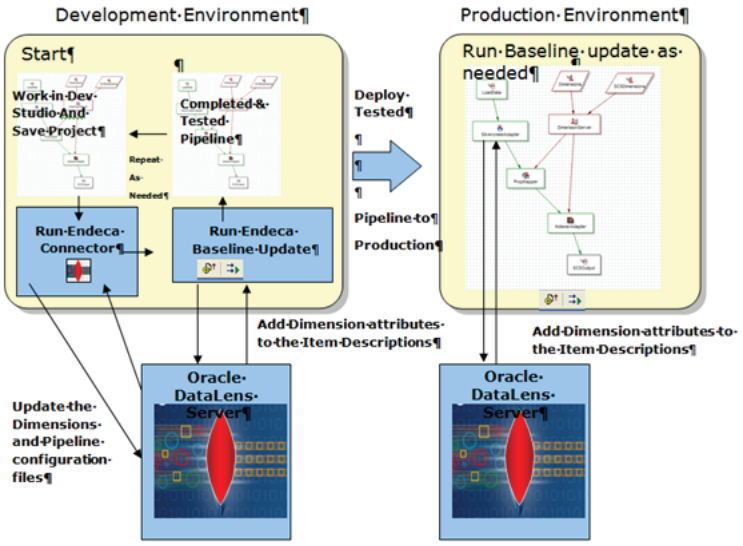
If it is set to `true`, then the Endeca Connector continues processing, even if it encounters errors (such as bad data, missing descriptions, etc.), and will not cause Forge to terminate processing.

# 6

# Using the Endeca Connector Adapter

This chapter explains how to use the Endeca Connector in Development and Production environments.

## Using the Endeca Connector Adapter




### Development Environment

The following are the steps for using the Endeca Connector Attribute Discovery and the Endeca Connector Adapter in a development/QA environment:

1. Endeca developers create a new project using the Endeca Developer Studio.
2. Developers add the Endeca Connector Adapter (Java Manipulator) to the Endeca Project.
3. Developers then run the Endeca Connector program to update the Endeca project.
4. Developers may re-open the project in the Endeca Developer Studio and create dimensions/properties/precedence rules are now part of the project.
5. Developers can run the Endeca Connector program independently of the Endeca Developer Studio if the data lens item definition attributes have been changed.

> **Note:** The project needs to be re-loaded to see the new attributes.

6. When the developers are done testing, the Endeca Project is sent to the production environment.

## Production Environment

The tested Endeca pipeline is deployed in the production environment.

Changes made to the data lens (using Knowledge Studio) should be checked into the development/QA environment for testing, prior to deployment to the Oracle DataLens Production Server Group for real-time processing.

When the Endeca baseline update is run on the production data, no additional steps are needed for the data lens processing. The processing has already been setup as part of the Endeca Pipeline as described in the previous chapters and requires no external intervention during production processing by Endeca.

# A

# Performance

This appendix describes performance aspects of interest.

## Oracle DataLens Servers

This section explains ways to improve performance on your Oracle DataLens Server when using the Endeca Connector.

### Tuning for the Endeca Connector Adapter Pass-Through Value

The Oracle DataLens Servers should be tuned to match the parameters used in the Endeca Connector Adapter.

By default, the Endeca Connector is set to send records from the Endeca Forge process to the Oracle DataLens Server in chunks of 15,000 records. Generally, the larger the chunk size, the greater the performance gain on record processing. The limit is the amount of memory allocated to the Forge process when running a baseline update on the Endeca ITL machine.

Assuming that you will leave the chunk size at 15,000 for the Endeca Connector Adapter, you do not need to change the default values of your Oracle DataLens Server.

You can view the default values for your server by:

1. Ensure that your Oracle DataLens Server is running.
2. Open one of the following supported Web browsers for your environment:
   - Internet Explorer 8.0 or later and Internet Explorer 9.0 or later
   - Mozilla Firefox 4.0 or later and Firefox 5.0 or later
   - Google Chrome 12.0 or later
   - Safari 5.0 or later
3. Enter the following URL:

   `http://`*`hostname:port`*`/datalens`

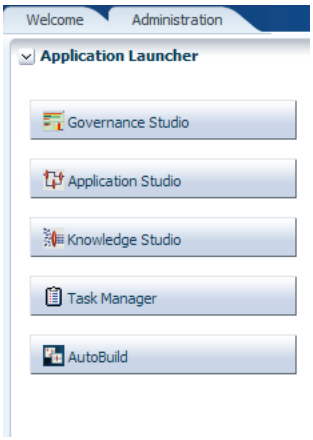
   where *hostname* is the DNS name or IP address of the Administration Server and *port* is the listen port on which the Administration Server is listening for requests (port 2229 by default).

   If you configured the Administration Server to use Secure Socket Layer (SSL) you must add s after http as follows:

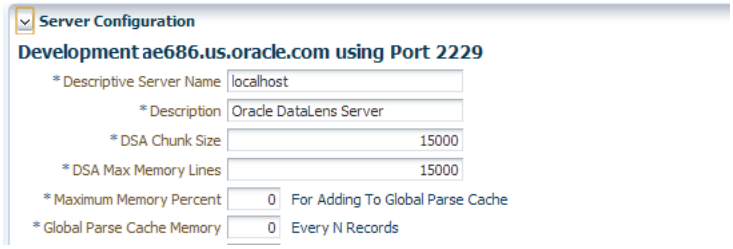
   `https://`*`hostname:port`*`/datalens`

4. When the login page appears, enter a user name and the password. Typically, this is the user name and password you specified during the installation process.

   The Oracle DataLens Server Web pages are displayed and default to the **Welcome** tab.



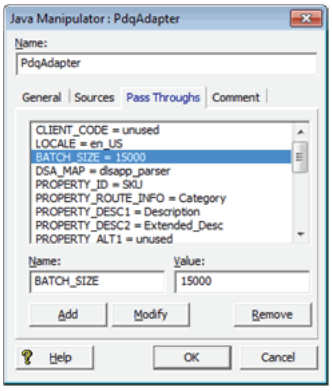

5. Select the **Administration** tab.
6. From the **Server** panel, select **Server Group.**
7. Click the link for your server.




   The values for the selected server are displayed.

   For additional information on these parameter, see *Oracle Enterprise Data Quality for Product Data Oracle DataLens Server Administration Guide*.

If you need to change the `BATCH_SIZE` parameter in the `PdqAdapter` (Java Manipulator) to a larger value, then the value on the Oracle DataLens Server should be increased to match this size.




You must restart your Oracle DataLens Server after changing any parameter values for them to be used.

## Multiple Production Servers

To speed up the processing of the data during the Forge processing by the `PdqAdapter`, multiple production servers can be put into a single Production server group. The Oracle DataLens Server will load balance the work of the data lens processing among all the Oracle DataLens Servers in the Production server group.

The main DSA that performs the processing by the `PdqAdapter` will use the Oracle DataLens Server's load-balancing and multi-threading capability to increase throughput when processing data on multiple servers with no additional configuration changes needed by the Endeca Connector administrator.

It should be noted that for the Endeca Connector "Discover/Delete" Transform Maps, the processing of the individual steps in a DSA are single-threaded, but the steps within a single Transform Map are multi-threaded. This means that when creating the Endeca Connector Discover Maps, each step should only perform a *single* Endeca Connector Add-In to prevent concurrent updating of the same Endeca project files by simultaneous job steps.

# Forge

There will be an overhead when the pipeline is being run during the Forge processing because the Oracle DataLens Server will be called to extract attributes for each line of input data. This processing will run quite fast, with processing speeds over 100 lines per second typical based on the input data and the quality of the Route Information. This means that when processing the input during the Forge step you would expect to add about two to seven minutes of processing for 40,000 lines of input data.

You can further speed the processing of this data by:

- Using Ultra-high priority DSA jobs.

  The smaller the chunk size, the more impact this will have on performance.

- Setting the Endeca Connector Adapter chunking levels higher for faster throughput.

  Performance improvements have e been observed from over 20 minutes for processing 40K records to under 2 minutes for processing the same records, simply by changing the chunk size (`BATCH_SIZE`) from 100 to 21000.

- Ensuring that the Route Information is accurate for the input data.

  This is important to minimize the number of data lenses that need to process each line of data.

- Running the processing on a fast Oracle DataLens Production server.

  Typically, the Production server is a more powerful machine than the Oracle DataLens Administration server, although the processing can take place on either type of server.

## Randomizing the Input Data in Forge

This is useful if you are using multiple parallel Oracle DataLens Servers to process the Forge data. If the data is randomized, then large performance gains are possible, especially if the data is grouped by product.

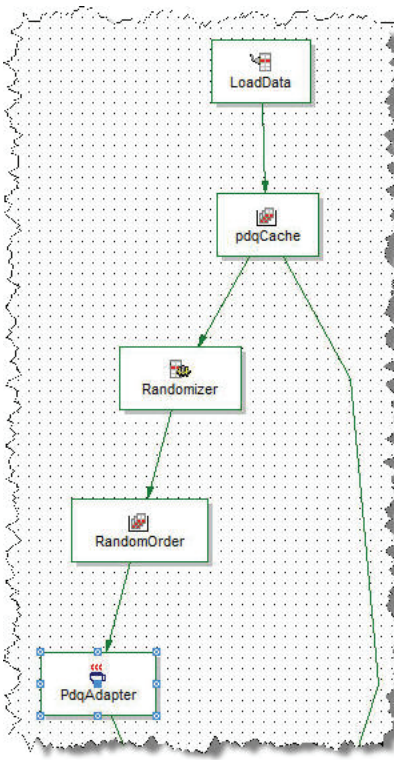
Here is a simple way to randomize the data in the pipeline. If you create a flow similar to the following diagram, you can randomize the data just before it goes through the `PdqAdapter`.

In the Perl manipulator, add a “next record” method with the following code:

```
# Add a random field
my $rec = $this->record_source(0)->next_record;

# Careful: $rec will be undefined if there are no more records.
if ($rec) {
   my $pval = new EDF::PVal("random", sprintf("%07d",int(rand(9999999))));
   $rec->add_pvals($pval);
}
return $rec;
```

In the `RandomOrder` cache, use 'random' as your record index.




## Dgidx

The Dgidx processing will also be expected to slow down. In this step, although there is no data being processing by the Oracle DataLens Server, the slowdown will be due to the additional Dimensions that are added to the input data and will need to be indexed for guided navigation and search.

If you double the number of Dimensions that are used by Endeca, then you would expect a corresponding increase in the processing time to index these additional attributes.

You can further speed the processing of the Dgidx step by spreading the Dgidx processing out over several Endeca servers

# B

# DSA Format

This appendix describes the format of the Endeca Connector DSA that will be called by the Endeca Connector Adapter; the DSA *must* be in format described.

## Inputs

- **ID** - The name of the ID field in the input data.
- **ROUTEINFO** - The "hint" used to more efficiently route the data.
- **DESC1** - The name of the first description field in the input data.
- **DESC2** - The name of the second description field in the input data.
- **ALT1** - First alternate data field (`mfgName`).
- **ALT2** - Second alternate data field (`mfgPartNo`).
- **ALT3** - Third alternate data field (user-defined).

The DSA need not use every one of these inputs if they are not needed. In fact, if only the Id and Description are needed (for testing for instance), then just pass an empty Route Info field through your DSA and just skip the other inputs past the 1st description. The DSA only extracts the number of fields that it needs.

Oracle recommends that exceptions be trapped in the DSA, rather than just dropping these records, so that you can route the records that were not processed to a location where they can be used to enhance the parsing and attribute extraction of the data lenses.

## Outputs

- **ID** - This is used by the Endeca Connector Adapter to associate the transformed data with the original data.
- **Return Value 1** - This information is passed back to Endeca
- **Return Value 2** - This information is passed back to Endeca
- **Return Value 3** - This information is passed back to Endeca
- **Key/Value pairs defining the attribute** - These are the attributes that are mapped to the Endeca Dimensions.

> **Note:** There is no Quality Index (QI) component in the input data, or in the output data. The QI checks can be performed in the DSAs to control the routing.

## Underscores and Spaces

The Endeca Connector Initialization programs will get the attribute names from the Item Definition Attribute Aliases in each of the data lenses. The Endeca Connector will then standardize all the alias names to be proper cased (if the `REPLACE_UNDERSCORES_ONLY` flag is set to `false`) and will always replace any of the underscores with spaces. This prevents the user from needing to match the case of all the aliases between all the item definitions in all the individual data lenses. This also makes the names with underscores more presentable to end users in the Web application (for example, Diagonal_Screen_Size to Diagonal Screen Size.)

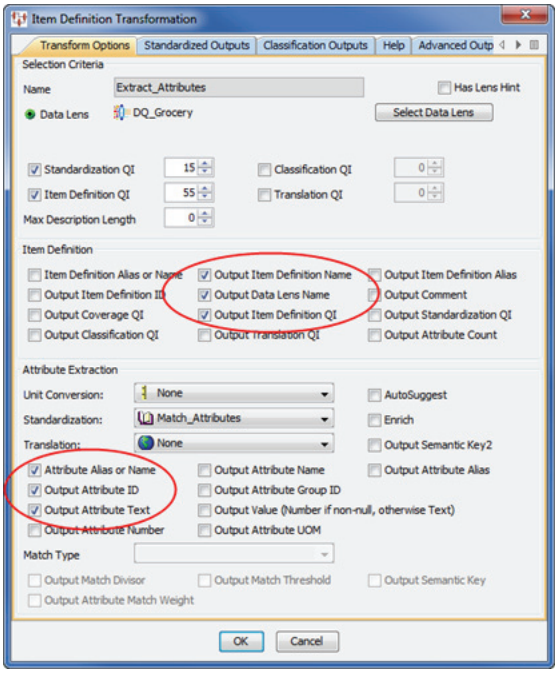
The Endeca Connector Adapter (`PdqAdapter` Java Manipulator as called from the Endeca Forge process), will also proper case (if the `REPLACE_UNDERSCORES_ONLY` flag is set to `false`), and will always replace underscores when processing the data.

## Attribute Aliases

The discovery process for the data lens attributes will always retrieve the attribute alias. If the attribute alias is not defined, then it will use the attribute name.

This means that when the Endeca Connector Adapter is running, it also needs to retrieve the attribute alias (if it exists) so that the Endeca Connector Discovery process and the Endeca Connector Adapter (run during the Forge processing) are in sync and will work properly.

This means that the following check box must be selected in your Transform Maps that are called by the main DSA used by the adapter.



# C

# Endeca Connector Robustness

The Endeca Connector supports high availability through:

- redundancy,
- round-robin Oracle DataLens Server support,
- and real-time fail-over of Oracle DataLens Servers during processing.

The Endeca Connector supports parallel processing and load balancing through:

- multiple parallel processing threads for each `PDQ_SERVER_`*n* defined
- and each thread fully supports the high availability

This is accomplished without the need for additional hardware support such as redundant clustered servers or intensive hardware support although these hardware solutions are fully supported. This reduces hardware infrastructure costs by having a very robust software solution. Additionally, it allows parallel processing, load balancing and high availability for the Endeca Connector Adapter when running as part of the Endeca Forge processing.

## Endeca Connector Redundancy

Redundancy is accomplished by having multiple Oracle DataLens Servers, all setup to process DSAs, and all setup to load and process the same data lenses. This is configured with the multiple Oracle DataLens Server configuration parameters supported by the Endeca Connector.

- `PDQ_SERVER_1 = DLFProdServerOne`
- `PDQ_SERVER_2 = DLFProdServerTwo`
- `PDQ_SERVER_3 = DLFProdServerThree`

These multiple redundant servers eliminate the need for additional hardware support for redundancy.

> **Note:** Three servers are defined in the example though there is no limit to the number of Oracle DataLens Servers that you can add.

These multiple redundant servers are used by both the Endeca Connector Add-In Discovery components (and the Deletion components) and the Endeca Connector `PdqAdapter`.

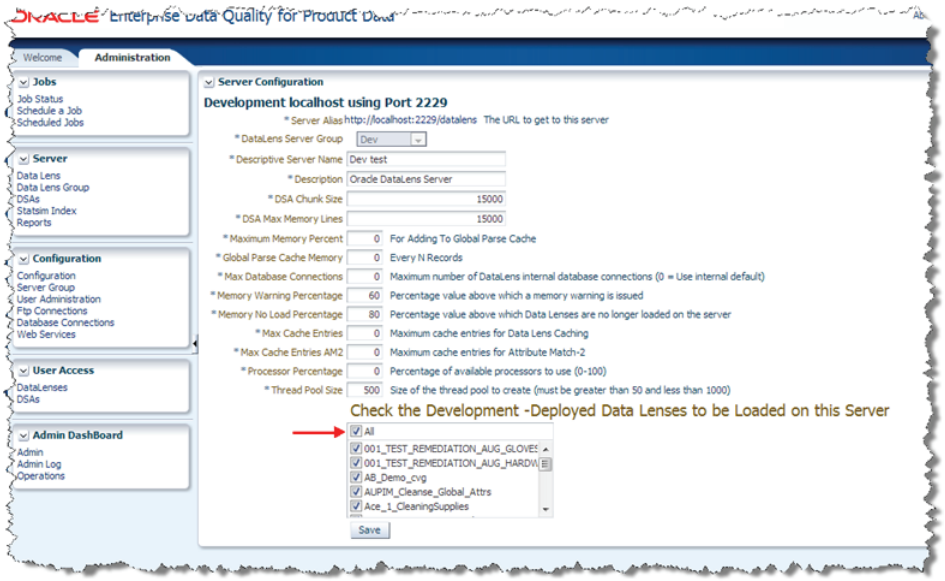
This solution is completely flexible and will work with almost any Oracle DataLens Server topology, such as the following:

- An Administration server and a Production server.
- An Administration server and multiple Production servers, all in the same server group.
- An Administration server and multiple Production servers, all in different server groups.
- Multiple production servers, all in the same server group. Oracle recommends this configuration.

## Configuring the DSA and Data Lens

The Endeca Connector DSA must be made available to *all* the Oracle DataLens Servers in any of the Development or Production Server Groups.

Each Oracle DataLens Server must have the **All** check box selected so that the all of the deployed data lenses used by the Forge process DSAs are loaded as in the following:




Go to the Oracle DataLens Server Administration web page and ensure this option is set for each Oracle DataLens Server in the appropriate Development and Production server groups. For more information, see *Oracle Enterprise Data Quality for Product Data Oracle DataLens Server Administration Guide*

## Round-Robin Support for Oracle DataLens Servers

The Endeca Connector DSA Transformation Add-Ins (the discovery processes) uses a round-robin approach to selecting an initial server for data processing. This means that the job will not even start until a Oracle DataLens Server is verified to be up and running. This is controlled by the `PDQ_SERVER` parameters that are set in the `PdqAdapter` pass through parameters and used by all the components of the Endeca Connector.

The round-robin checking always starts with `PDQ_SERVER_1` and then checks `PDQ_SERVER_2` and finally `PDQ_SERVER_3`.

Note that the Endeca Connector Adapter is more sophisticated and keeps track of the last accessed server when doing the round-robin fail-over.

The Oracle DataLens Servers all have a "ping servlet" so that the Endeca Connector can ensure not only that the server is running, but also that the Oracle DataLens Server service is running and processing requests.

### Add-In Transforms

Following is an example of the round-robin server connection from the Oracle DataLens Server log file for a "Discover Precedence" DSA job.

```
INFO 16 Sep 2008 15:53:07 [] -  PDQ-Endeca Connector Dimension Discovery Version
11.1.1.6.0, Build 20120804 Copyright (c) 2012, 2012, Oracle and/or its affiliates.
All rights reserved.
INFO 16 Sep 2012 15:53:10 [] - Attempted 0 times to connect to http://
DLFProdServerOne:2229/datalens/Ping
ERROR 16 Sep 2012 15:53:10 [] - Failed to connect to server (http://
DLFProdServerOne:2229/datalens/Ping)[PingRequest]: DLFProdServerOne
INFO 16 Sep 2012 15:53:10 [] - PdqAdapter parameters:
    DSA_MAP = endeca_demo_dimensions
    REPLACE_UNDERSCORES_ONLY = true
    USE_PDQ_rPREFIX = false
    Using PDQ_SERVER_2 (DLFProdServerTwo:2229)
INFO 16 Sep 2012 15:53:10 [] - Connecting to Server DLFProdServerTwo and port 2229
INFO 16 Sep 2012 15:53:10 [] - Running on  DataLens Admin server
DLFProdServerTwo:2229
```

This failed to get a response from the `DLFProdServerOne` and ended up connecting to the `DLFProdServerOne`. The log also reports on which `PDQ_SERVER` is being used.

## Endeca Connector Fail-over

The Endeca Connector fail-over is a component that works when processing the actual data with the `PdqAdapter` during the Endeca Forge processing. This is optimized over a hardware fail-over solution because the Endeca Connector Fail-over will resubmit the data chunk to an alternate server is a problem is encountered, continuing the Forge processing. If a job is processing chunk 15 of a total of 20 chunks, the fail-over will resubmit data chunk 15 to a redundant Oracle DataLens Server, continuing the Forge processing without Forge ever being aware that a Oracle DataLens Server went down.

A hardware fail-over will require that the Forge job is re-submitted from the start.

The fail-over will occur if the following occur:

- The DSA Job has a fault and fails to respond.
- The DSA machine has any type of connection error such as the server hardware failure or Tomcat failure.
- The DSA machine has a memory error such as a Java heap space error.

> **Note:** The Endeca Connector Adapter keeps track of the last accessed server among all the servers defined when doing the round-robin fail-over and will use this information to determine which server to send a chunk of data to for re-processing.

Here is the result of pulling the plug on one of the Oracle DataLens Servers:

```
Endeca51:2229-2 2009.02.04_03:51:00 Running a data chunk on the DLS Server
Endeca51:2229
Endeca51:2229-2 2009.02.04_03:51:01 Processing a chunk of 9950 records on the
```

```
Endeca51:2229 DLS server
Endeca51:2229-2 2009.02.04_03:51:31 DLF Server Endeca51:2229 is not responding
Endeca51:2229-2 2009.02.04_03:51:31 Warning: Caught a Connection Exception, trying
another server...
Endeca51:2229-2 2009.02.04_03:52:13 DLF Server Endeca51:2229 is not responding
Endeca51:2229-2 2009.02.04_03:52:13 Retrying the chunk with the DL Server
admin1-M6300:2229
Endeca51:2229-2 2009.02.04_03:52:14 Processing a chunk of 9950 records on the
admin1-M6300:2229 DLS server
Endeca51:2229-2 2009.02.04_03:52:15 Using Job Id: 182
Endeca51:2229-2 2009.02.04_03:53:04 Job#182 DLS Server returned 7600 records from
the chunk
```

In the preceding example, PDQ_SERVER_1 is pinged to verify that there was just not a network issue. Then the server is hot-swapped to PDQ_SERVER_2 and the entire chunk is re-submitted. The last line in the preceding log snippet is the first line of re-submitted data for this current chunk.

The following error message will be output to the log file if the WebLogic Server is stopped or fails:

```
admin1-M6300:2229-1 2009.02.04_03:45:07 Warning: Caught a Job Failed Fault, trying
another server...
```

## Example of Single Threaded Versus Multiple Oracle DataLens Servers

This first example is of the Endeca Connector Adapter running with a single Oracle DataLens Server.

All the data chunks are being processed in parallel threads (one per server) on three separate Oracle DataLens Servers.

```
PDQ-Endeca Connector Adapter (Endeca Java Manipulator) Version 11.1.1.6.0, Build
20120804 Copyright (c) 2008, 2012, Oracle and/or its affiliates. All rights
reserved.
PDQ_SERVER_1 - Adding required DataLens Server cwellell-M6300:2229
PDQ_SERVER_2 - Adding optional High-Availability, Load-Balanced,
Parallel-Processing DataLens Server Endeca51:2229
PDQ_SERVER_3 - Adding optional High-Availability, Load-Balanced,
Parallel-Processing DataLens Server cwellell-VM:2229
cwellell-M6300:2229-1 2009.02.12_01:10:07.000 Running a data chunk on the DLS
Server cwellell-M6300:2229
cwellell-M6300:2229-1 2009.02.12_01:10:07.343 Processing a chunk of 10000 records
on the cwellell-M6300:2229 DLS server
cwellell-M6300:2229-1 2009.02.12_01:10:08.796 Using Job Id: 208
cwellell-M6300:2229-1 2009.02.12_01:10:23.031 Job#208 DLS Server returned 8965
records from the chunk

cwellell-VM:2229-3 2009.02.12_01:10:12.890 Running a data chunk on the DLS Server
cwellell-VM:2229
cwellell-VM:2229-3 2009.02.12_01:10:13.984 DLF Server cwellell-VM:2229 is not
responding
cwellell-VM:2229-3 2009.02.12_01:10:14.125 Retrying the chunk with the DL Server
Endeca51:2229
cwellell-VM:2229-3 2009.02.12_01:10:14.359 Processing a chunk of 3041 records on
the Endeca51:2229 DLS server
cwellell-VM:2229-3 2009.02.12_01:10:20.859 Using Job Id: 10
cwellell-VM:2229-3 2009.02.12_01:11:17.343 Job#10 DLS Server returned 1954 records
from the chunk

Endeca51:2229-2 2009.02.12_01:10:11.250 Running a data chunk on the DLS Server
```

```
Endeca51:2229
Endeca51:2229-2 2009.02.12_01:10:11.921 Processing a chunk of 9950 records on the
Endeca51:2229 DLS server
Endeca51:2229-2 2009.02.12_01:10:43.515 Using Job Id: 11
Endeca51:2229-2 2009.02.12_01:11:06.500 Job#11 DLS Server returned 7600 records
from the chunk

********* Processed 3 Chunks  with 22991 total input lines *********
********* Updated   18519 total lines by the PDQ-Endeca Connector *********
********* Completed the PDQ-Endeca Connector processing in 73 seconds *********
```

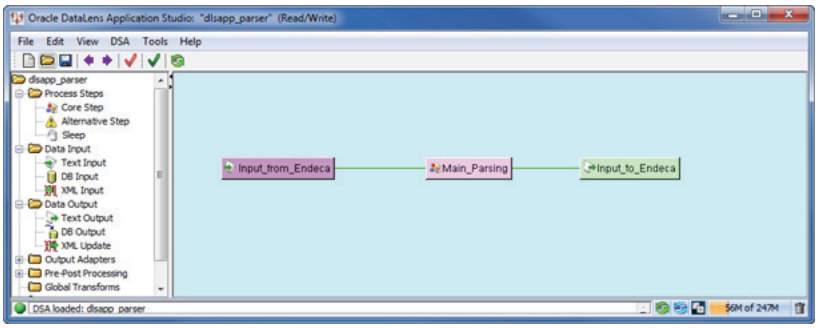
# D

# Setting Up an Example Endeca Connector Project

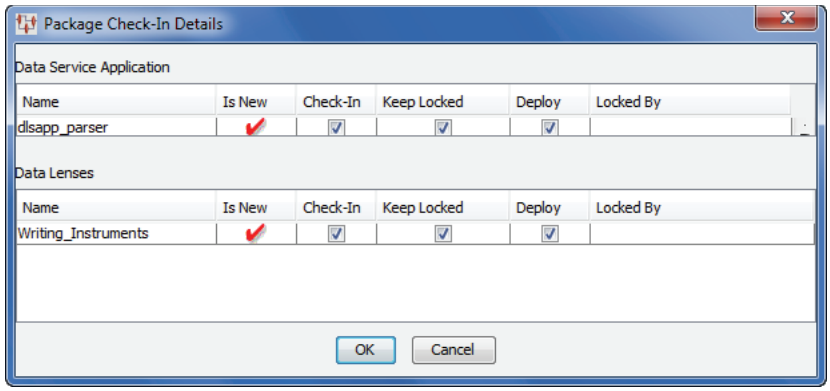
The 'dlsapp' project is a fully self-contained Endeca Connector project and DSA package that you can copy into place to illustrate how the Endeca Connector Adapter works within the Endeca pipeline. This appendix explains how to use the example files included in the `opdq-connector-endeca` directory.

First, you extract the example files and check-in the DSA package in the Application Studio:

1. On your Endeca server, go to the `edqp` directory you created during installation.
2. Change directories to `exampleProject` directory.
3. Extract all of the files in the dlsapp.zip into this directory.
4. Start the Application Studio.
5. From the **File** menu, select **Import Package**.
6. Select the **dlsapp_parser.pmap** DSA package from the `exampleProject` directory and click **OK**.




7. From the **DSA** menu, select **Check-In Package** so that you can check in this DSA and the data lens associated with it at one time.




8. Ensure that **Writing_Instruments** data lens is selected for check in and click **OK**. A progress dialog box appears so that you can view the check ins as they occur and details any errors found.

Next, you create a project to setup the necessary structure and then replace it with the Endeca Connector 'dlsapp' project:

1. Open a Windows Command Prompt (`cmd.exe`).
2. Change directories to `C:\Endeca\Solutions\deploymentTemplate-3.2\bin`.
3. Enter **deploy.bat** and press **Enter** to create the project structure.
4. Enter the following values to the program prompts:

| Program Prompt | Response Value |
|---|---|
| Deployment type: | **1 (Dgraph deployment)** |
| Application name: | **dlsapp** |
| Deployment directory: | **C:\Endeca\applications** |
| EAC port: | **8888** |
| Enable IAP Workbench integration: | **Y** |
| IAP Workbench port: | **8086** |
| Port for Dygraph1: | **15000** |
| Port for Dygraph2: | **15002** |

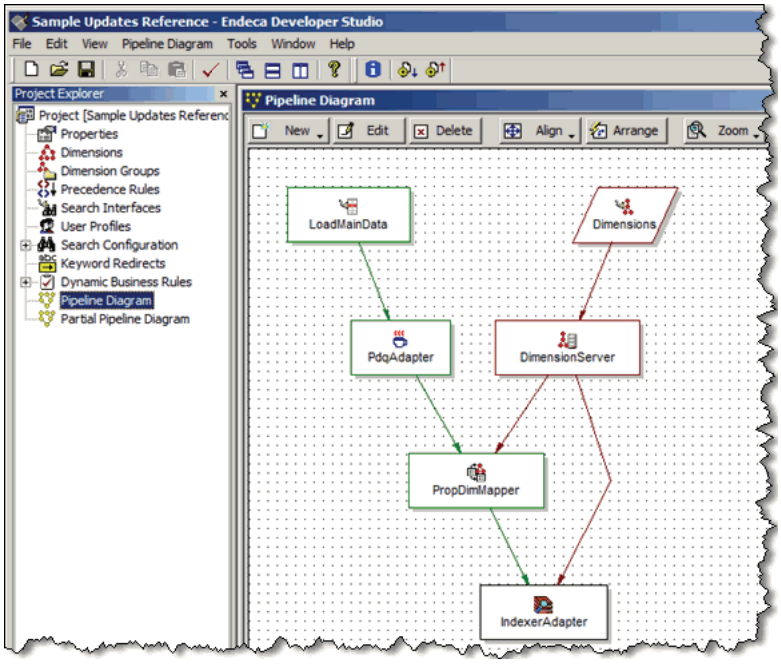
Now, an empty version of a dlsapp project has been created in the Development Studio directory and makes that project known to the Endeca server, which is very important. Next, you replace this initial dlsapp project version with the Endeca Connector pre-populated project.

5. Change directories to `C:\Endeca\DeveloperStudio\projects`.
6. Delete `dlsapp` directory.
7. To replace the initial dlsapp project version, you create a new copy of the `dlsapp` folder with all of the project values in place. Copy the `dlsapp.zip` into the `projects` folder and extract all files from it.

**Note:** The `dlsapp.zip` file is contained in the Endeca Connector file, `opdq-endeca-connector.zip` that is installed as part of the EDQP11g R1 (11.1.1.6.1) release.

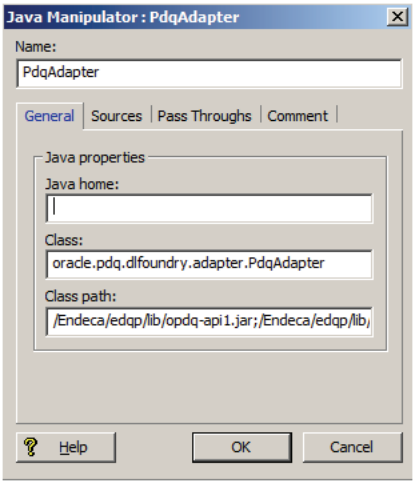
Next, confirm that the project is available for use.

8. Start the Endeca Developer Studio.
9. From the **File** menu, select **Open**.
10. Locate the `C:\Endeca\DeveloperStudio\projects\scstrain\config\pipeline` directory and select the **dlsapp.esp** project to open it.
11. Select **Pipeline Diagram** from the Project Explorer options to inspect the structure of the dlsapp pipeline.



**Note:** For Linux, you must open the PdqAdapter then change the Class path separator from a semi-colon ';' as the separator between the paths for each Endeca Connector jar file to a colon ':' as the separator as in the following:

```
/Endeca/edqp/lib/opdq-api1.jar;/Endeca/edqp/lib/jdom-1.0.jar
;/Endeca/edqp/lib/opdq-core.jar;/Endeca/edqp/opdq-connector-
endeca.jar
```




For more information, see "Adding the Endeca Connector Adapter" on page 3-1.

This pipeline diagram is a very simple one though they can get far more complex. The purpose is to illustrate the placement of the PdqAdapter in the overall pipeline flow and provide an example of the configuration necessary for the Endeca Connector to operate correctly.

# E

# Endeca Connector Troubleshooting

Troubleshooting the loader and pipeline processes is typically straightforward. The Oracle DataLens Server is verbose about the errors it encounters, and typically the error messages indicate how to locate and correct them. There is an additional level of logging available by turning on the `PDQ.Tracing` parameter in the discovery processes, which is very verbose about the data lens each attribute comes from, and what dimensions were discovered. If the process completes successfully, you can look in the Endeca Developer Studio, and see that the dimensions have been added and that precedence rules have been built to the appropriate parent dimensions.

The Endeca Connector is configured through a number of pass through parameters as follows: The Endeca Connector parameters are:

| | |
|---|---|
| `BATCH_SIZE` | The batch size property controls how much data is sent to the server(s) as a chunk. The chunk data is held in memory on the Endeca Server. While increasing the chunk size can improve performance, if the data fills the Endeca buffers, this causes a fatal error. The default chunk size is 15000, a recommend maximum would be 20000 and of course depends on your hardware configuration. |
| `DSA_MAP` | The DSA which will be called on the Oracle DataLens Server to process the sent data. This is also used by the loader process as described above. |
| `PROPERTY_ID` | The property to be used as the first (id) field in the DSA. If this property is not available for a record, errors can be generated. |
| `PROPERTY_ROUTE_INFO` | The second field in the DSA. This property will be used to route the data to the correct data lens. |
| `PROPERTY_DESC_1`<br>`PROPERTY_DESC_2` | The fields with the record descriptions. While not required, if you do not provide any description, the results will not be very useful. |
| `PROPERTY_ALT1`<br>`PROPERTY_ALT2`<br>`PROPERTY_ALT3` | Optional fields which can provide additional fields to the DSA. These can change the behavior of the process, or include additional information for standardization or routing. |
| `RETURN_VAL1`<br>`RETURN_VAL2`<br>`RETURN_VAL3` | Required fields returned from the DSA. These are typically used for data lens transform fields, such as Item Definition Quality or Standardized Description. These fields do not have an attribute name/value pair, so this pass through is used to assign a property name to the returned value. |
| `PDQ_SERVER`*n* | The server name and port number for the first processing server. Multiple servers can be defined. This allows for fail over and load balancing capabilities for large installations. |

Optional parameters can be disabled by setting them to the string `unused`.

Troubleshooting the pipeline process falls into three main categories, based on the log file you need to examine as described in the following:

| | |
|---|---|
| Forge errors | Review the `Forge.log`. Forge is reasonably good about what caused the error. If forge indicates the problem is with the `PdqAdapter`, then you need to review the `Edf.Pipeline.RecordPipeline.JavaManipulator.PdqAdapter.log`. |
| Oracle DataLens Server errors and Endeca pipeline errors | Review the `Edf.Pipeline.RecordPipeline.JavaManipulator.PdqAdapter.log` file. This log file contains the transactions between the Endeca pipeline and the Oracle DataLens Server. This is the main log used to troubleshoot the EDQP Endeca Connector integration. It may not be accurate in its errors because it does not have a lot of visibility into what has happened on the Endeca or EDQP sides. Additional logging can be enabled with the `PDQDEBUG` pass through parameter; this debug information is very verbose so it can fill your disks quickly if you leave it on. |
| Oracle DataLens Server errors and information | Review the `dataserver.log` file. Go to the Oracle DataLens Server Administration web page and review all job status information. For more information, see *Oracle Enterprise Data Quality for Product Data Oracle DataLens Server Administration Guide*.<br><br>Typically, you will be able to isolate the cause of an error in the DSA process. It is also useful if an error does not occur though no results are produced. The `dataserver.log` on the Oracle DataLens Server contains verbose errors, including stack traces if configured. |