

# Oracle Commerce Guided Search Platform Services

XML Reference

Version 11.1 • July 2014





# Contents

<b>Preface</b> .....	<b>9</b>
About this guide.....	9
Who should use this guide.....	9
Conventions used in this guide.....	9
Contacting Oracle Support.....	10
<b>Chapter 1: Endeca XML Reference</b> .....	<b>11</b>
Welcome.....	11
Analytics_config.xml elements.....	11
Common.dtd elements.....	11
COMMENT.....	11
compression.....	12
CONFIG.....	12
DIMENSION_ID.....	13
DIMNAME.....	14
DVAL_ID.....	14
DVAL_PATH.....	15
KEY_DIMENSION.....	16
LOG.....	17
PHRASE.....	18
PROFILE.....	18
PROP.....	19
PROPNAME.....	19
PVAL.....	20
RECORD_GROUP.....	20
RECORD_ID.....	20
RECORD_INDEX.....	21
Derived_props.xml elements.....	22
DERIVED_PROP.....	22
DERIVED_PROPS.....	22
Dimensions.xml elements.....	23
BOUND.....	23
DIMENSION.....	24
DIMENSIONS.....	25
DIMENSION_NODE.....	26
DVAL.....	27
LBOUND.....	28
NUM_DVALS.....	29
SYN.....	30
UBOUND.....	31
Dimension_groups.xml elements.....	31
DIMENSION_GROUP.....	32
DIMENSION_GROUPS.....	32
Dimension_refs.xml elements.....	33
DIMENSION_REF.....	33
DIMENSION_REFS.....	34
Dimsearch_index.xml elements.....	35
DIMSEARCH_HIERARCHY.....	35
DIMSEARCH_INDEX.....	36
Dval_ranks.xml elements.....	38
DVAL_RANK.....	38
DVAL_RANKS.....	39
Dval_refs.xml elements.....	40
DVAL_REF.....	40
DVAL_REFS.....	41
Externaldimension elements.....	42
Key_props.xml elements.....	42
Languages.xml elements.....	42

KEY_LANGUAGE.....	42
LANGUAGES.....	43
Phrases.xml elements.....	44
PHRASES.....	44
PHRASE_ADDITIONS.....	45
DIMENSION_IMPORTS.....	45
Pipeline.epx elements.....	46
DIMENSION_ADAPTER.....	46
DIMENSION_SERVER.....	48
DIMENSION_SOURCE.....	50
DVAL_ID_LIMITS.....	51
EXPRBODY.....	51
EXPRESSION.....	52
EXPRNODE.....	54
ID_SERVER.....	55
INDEXER_ADAPTER.....	56
JAVA_MANIPULATOR.....	58
JOIN_ENTRY.....	60
PASS_THROUGH.....	60
PERL_MANIPULATOR.....	63
PERL_METHOD.....	65
PIPELINE.....	67
PROP_MAPPER.....	69
PROP_MAPPING_DIM.....	71
PROP_MAPPING_NONE.....	73
PROP_MAPPING_PROP.....	73
RECORD_ADAPTER.....	75
RECORD_ASSEMBLER.....	80
RECORD_CACHE.....	82
RECORD_JOIN.....	83
RECORD_MANIPULATOR.....	84
RECORD_SOURCE.....	85
ROLLOVER.....	86
TRANSFORMER.....	88
UPDATE_ADAPTER.....	89
Dimsearch_config.xml elements.....	91
DIMSEARCH_CONFIG.....	91
Precedence_rules.xml elements.....	92
PRECEDENCE_RULE.....	92
PRECEDENCE_RULES.....	93
Profiles.xml elements.....	94
PROFILES.....	94
Project.xml elements.....	95
DIMENSION_INPUTS.....	95
INPUT.....	95
PROJECT.....	96
RECORD_INPUTS.....	97
Prop_refs.xml elements.....	98
PROP_REF.....	98
PROP_REFS.....	99
Record_filter.xml elements.....	100
RECORD_FILTER.....	100
Record_id_prop.xml elements.....	101
RECORD_ID_PROP.....	101
Record_sort_config.xml elements.....	102
RECORD_SORT.....	102
RECORD_SORT_CONFIG.....	103
Record_spec.xml element.....	103
RECORD_SPEC.....	104
Recsearch_config.xml elements.....	104
RECSEARCH_CONFIG.....	105
Recsearch_indexes.xml elements.....	106
RECSEARCH_INDEX.....	107
RECSEARCH_INDEXES.....	107

Refinement_config.xml elements.....	110
CLUSTERS.....	110
DYNAMIC_RANKING.....	111
REFINEMENTS.....	112
REFINEMENTS_CONFIG.....	114
STATS.....	115
Relrank_strategies.xml elements.....	116
RELRANK_APPROXPHRASE.....	116
RELRANK_EXACT.....	117
RELRANK_FIELD.....	117
RELRANK_FIRST.....	118
RELRANK_FREQ.....	118
RELRANK_GLOM.....	119
RELRANK_INTERP.....	119
RELRANK_MAXFIELD.....	120
RELRANK_MODULE.....	120
RELRANK_NTERMS.....	121
RELRANK_NUMFIELDS.....	121
RELRANK_PHRASE.....	122
RELRANK_PROXIMITY.....	123
RELRANK_SPELL.....	123
RELRANK_STATIC.....	124
RELRANK_STRATEGIES.....	125
RELRANK_STRATEGY.....	125
RELRANK_WFREQ.....	127
Render_config.xml elements.....	128
RENDER.....	128
RENDER_CONFIG.....	129
Rollups.xml elements.....	130
ROLLUP.....	130
ROLLUPS.....	130
Search_chars.xml elements.....	131
SEARCH_CHAR.....	131
SEARCH_CHARS.....	132
Searchindex.xml elements.....	132
POSITIONAL_INDEX.....	133
SEARCH_INDEX.....	133
WILDCARD_INDEX.....	134
Search_interface.xml elements.....	135
AUTO_SUGGEST.....	135
DID_YOU_MEAN.....	136
MEMBER_NAME.....	136
PARTIAL_MATCH.....	137
SEARCH_CORRECTION_PARAMETERS.....	138
SEARCH_INTERFACE.....	139
Spell_config.xml elements.....	141
ASPELL.....	141
DICT_PER_LANGUAGE.....	142
ESPELL.....	143
FIRST_DICT.....	144
SPELL_CONFIG.....	145
SPELL_ENGINE.....	146
UNION_DICTS.....	147
Stemming.xml elements.....	148
STEM_CS.....	148
STEM_DA.....	148
STEM_DE.....	148
STEM_EL.....	149
STEM_EN_UK.....	149
STEM_EN_US.....	149
STEM_ES.....	150
STEM_FR.....	151
STEM_HU.....	152
STEM_IT.....	152

STEM_JP.....	153
STEM_KO.....	153
STEM_NL.....	154
STEM_PL.....	155
STEM_PT.....	155
STEM_XX.....	156
STEM_ZH_CN.....	156
STEM_ZH_TW.....	157
Stop_words.xml elements.....	157
STOP_WORD.....	158
STOP_WORDS.....	158
Studio_project.esp elements.....	159
PROJECT_FILE.....	159
PROJECT_MANAGER_CONFIG.....	160
PROJECT_PIPELINE.....	161
STUDIO_PROJECT.....	161
Thesaurus.xml elements.....	163
THESAURUS.....	163
THESAURUS_ENTRY.....	164
THESAURUS_ENTRY_ONEWAY.....	164
THESAURUS_FORM.....	165
THESAURUS_FORM_FROM.....	166
THESAURUS_FORM_TO.....	166

---

## Copyright and disclaimer

Copyright © 2003, 2014, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

---

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

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

# Preface

Oracle Commerce Guided Search is the most effective way for your customers to dynamically explore your storefront and find relevant and desired items quickly. An industry-leading faceted search and Guided Navigation solution, Guided Search enables businesses to influence customers in each step of their search experience. At the core of Guided Search is the MDEX Engine™, a hybrid search-analytical database specifically designed for high-performance exploration and discovery. The Oracle Commerce Content Acquisition System provides a set of extensible mechanisms to bring both structured data and unstructured content into the MDEX Engine from a variety of source systems. The Oracle Commerce Assembler dynamically assembles content from any resource and seamlessly combines it into results that can be rendered for display.

Oracle Commerce Experience Manager enables non-technical users to create, manage, and deliver targeted, relevant content to customers. With Experience Manager, you can combine unlimited variations of virtual product and customer data into personalized assortments of relevant products, promotions, and other content and display it to buyers in response to any search or facet refinement. Out-of-the-box templates and experience cartridges are provided for the most common use cases; technical teams can also use a software developer's kit to create custom cartridges.

## About this guide

This reference provides descriptions of the XML elements in Endeca project files, including the XML elements used to build components for a Forge pipeline.

It assumes that you are familiar with Endeca Developer Studio, with the concepts of the Endeca Information Transformation Layer, and with the configuration of the features of the Endeca MDEX Engine (such as search interfaces and dimensions). The general structure of the reference is based on the various XML files that constitute an Endeca project.

## Who should use this guide

This guide is intended for application developers who are building Endeca applications. In particular, it is intended as a reference for developers who must work with the XML files that define an application.

## Conventions used in this guide

This guide uses the following typographical conventions:

Code examples, inline references to code elements, file names, and user input are set in `monospace` font. In the case of long lines of code, or when inline monospace text occurs at the end of a line, the following symbol is used to show that the content continues on to the next line: `↵`

When copying and pasting such examples, ensure that any occurrences of the symbol and the corresponding line break are deleted and any remaining space is closed up.

## Contacting Oracle Support

Oracle Support provides registered users with answers to implementation questions, product and solution help, and important news and updates about Guided Search software.

You can contact Oracle Support through the My Oracle Support site at <https://support.oracle.com>.

## Chapter 1

---

# Endeca XML Reference

This reference describes the XML elements contained in the .xml files for an Endeca project.

## Welcome

This reference describes the XML elements in Endeca project files. The reference documents each element's DTD, attributes, sub-elements, and provide an example of its usage.

When you create an Endeca project using Developer Studio, the application creates the .xml files for you. As you modify your project, Developer Studio and Oracle Endeca Workbench write your changes to the .xml files. This reference explains how to configure XML elements if you choose to modify the .xml files directly.

Customers should use Developer Studio or Oracle Endeca Workbench to configure a project and only modify the XML directly when required in unusual exceptions.

## Analytics\_config.xml elements

The Analytics\_config.xml file is obsolete.

## Common.dtd elements

Common.dtd defines common elements that are available for use in multiple Endeca XML files.

## COMMENT

The COMMENT element associates a comment with a pipeline component and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form `<!-- ... -->`.

### DTD

```
<!ELEMENT COMMENT (#PCDATA)>
```

### Attributes

The COMMENT element has no attributes.

## Sub-elements

The COMMENT element has no sub-elements.

## Example

This example includes an informational comment.

```
<RECORD_ADAPTER URL="../incoming/wine_data.txt" NAME="LoadMainData" FORMAT="DE-
LIMITED"
DIRECTION="INPUT" FRC_PVAL_IDX="TRUE" COL_DELIMITER="|" ROW_DELIMITER="|\n" FIL-
TER_EMPTY_PROPS="TRUE">
  <COMMENT>This is the primary record adapter in the pipeline.</COMMENT>
</RECORD_ADAPTER>
```

## compression

The compression entity controls the level of data compression. The entity declares a shortcut for the enumeration of possible COMPRESSION\_LEVEL values. In the DTD, this entity is declared as % compression; however, in XML files, this entity is expanded to its full form as an attribute value COMPRESSION\_LEVEL="...".

The default compression level for this attribute is 0 (no compression). The value -1 for this attribute has been deprecated. Values 1-9 indicate increasing levels of compression.



**Note:** Disregard parameter entities that appear in Common.dtd. They are set by the system and their values should not be modified.

## DTD

```
<!ENTITY % compression "COMPRESSION_LEVEL (-1 | 0 | 1 | 2 | 3
| 4 | 5 | 6 | 7 | 8 | 9) #IMPLIED">
```

## Example

This example shows an Indexer Adapter writing compressed records.

```
<INDEXER_ADAPTER COMPRESSION_LEVEL="7" FILTER_UNKNOWN_PROPS="TRUE"
  FRC_PVAL_IDX="TRUE" MULTI_PROP_NAME=""
  NAME="IndexerAdapter" OUTPUT_DIMENSION_FORMAT="XML"
  OUTPUT_PREFIX="wine" OUTPUT_RECORD_FORMAT="BINARY"
  OUTPUT_URL="../partition0/forge_output/">
  <RECORD_SOURCE>PropDimMapper</RECORD_SOURCE>
  <DIMENSION_SOURCE>DimensionServer</DIMENSION_SOURCE>
  <ROLLOVER CUTOFF="2000000000" NAME="RECORD"
    NUM_IDX="1" PROP_NAME="" PROP_TYPE="ALPHA"
    REMOVE_PROP="FALSE" ROLL_URL=""
    TYPE="SIZE" VALIDATE="FALSE"/>
</INDEXER_ADAPTER>
```

## CONFIG

The CONFIG element provides generic configuration information as name/value pairs.

When used within a RECORD\_JOIN element, CONFIG provides flags to adjust the behavior of join operations. The NAME attribute specifies the type of configuration flag, and the VALUE attribute either enables or disables the flag. For more information about joining source data, see the Endeca Developer Studio Help.

**DTD**

```
<!ELEMENT CONFIG EMPTY>
<!ATTLIST CONFIG
  NAME CDATA #REQUIRED
  VALUE CDATA #REQUIRED
>
```

**Attributes**

The following sections describe the CONFIG element's attributes.

**NAME**

Provides the name of a configuration option. There are three valid NAME values:

- **MULTI\_SUB\_RECORDS** Valid only for left joins when there are multiple values for a given key dimension. If the record from the Left source has multiple values for a key dimension, the values are to be OR'd (creating multiple record keys). For a record index with a single key dimension this means that each value of that key dimension is an independent record key.
- **TRACK\_PROP\_SOURCE** If this attribute is TRUE, a new property is added to the joined record that tracks the record source from which the property value originated. The value of the new property is a combination of the record source name, the property name, and property value. The default value of TRACK\_PROP\_SOURCE is FALSE.
- **UNIQUE\_RECORD\_PROPS** If this attribute is TRUE, duplicate property values (same name, value and children) are not added when records are combined. The property values of the new record are unique. The default value of UNIQUE\_RECORD\_PROPS is FALSE. See "Joining Source Data" in the Developer Studio help for more details about each NAME value.

**VALUE**

Provides a value that enables or disables the NAME configuration option. The valid values are TRUE or FALSE.

**Sub-elements**

The CONFIG element has no sub-elements.

**Example**

In this example, several name/value pairs are established as part of join processing.

```
<RECORD_JOIN JOIN_TYPE="OUTER_JOIN">
  <CONFIG NAME="MULTI_SUB_RECORDS" VALUE="FALSE"/>
  <CONFIG NAME="TRACK_PROP_SOURCE" VALUE="FALSE"/>
  <CONFIG NAME="UNIQUE_RECORD_PROPS" VALUE="FALSE"/>
  <JOIN_ENTRY>
    <RECORD_SOURCE>c1</RECORD_SOURCE>
    <KEY_DIMENSION TYPE="PROPERTY_VALUE" ID="record_id"/>
  </JOIN_ENTRY>
  <JOIN_ENTRY>
    <RECORD_SOURCE>c2</RECORD_SOURCE>
    <KEY_DIMENSION TYPE="PROPERTY_VALUE" ID="id"/>
  </JOIN_ENTRY>
</RECORD_JOIN>
```

**DIMENSION\_ID**

The DIMENSION\_ID element specifies the ID number of a dimension.

**DTD**

```
<!ELEMENT DIMENSION_ID EMPTY>
<!ATTLIST DIMENSION_ID
  ID      CDATA      #REQUIRED
>
```

**Attributes**

The DIMENSION\_ID element has no attributes.

**Sub-elements**

The DIMENSION\_ID element has no sub-elements.

**Example**

In this example, a DIMENSION\_REF element uses a DIMENSION\_ID element to identify the dimension in question.

```
<DIMENSION_REF TYPE="SECONDARY" DIM_SEARCH_HIERARCHY="FALSE" RECORD_SEARCH_HIERARCHY="FALSE">
  <DIMENSION_ID ID="12"/>
</DIMENSION_REF>
```

**DIMNAME**

The DIMNAME element specifies the name of a dimension. This element provides an alternative to identifying a dimension by its ID with DIMENSION\_ID.

**DTD**

```
<!ELEMENT DIMNAME (#PCDATA)>
```

**Attributes**

The DIMNAME element has no attributes.

**Sub-elements**

The DIMNAME element has no sub-elements.

**Example**

This example creates a dimension group with three dimensions identified by their DIMNAME element.

```
<DIMENSION_GROUP NAME="Features">
  <DIMNAME>Number of speakers</DIMNAME>
  <DIMNAME>Horizontal resolution</DIMNAME>
  <DIMNAME>Vertical resolution</DIMNAME>
</DIMENSION_GROUP>
```

**DVAL\_ID**

The DVAL\_ID element specifies a dimension value ID.

**DTD**

```

<!ELEMENT DVAL_ID EMPTY>
<!ATTLIST DVAL_ID
  DIMENSION_ID  CDATA    #IMPLIED
  ID            CDATA    #REQUIRED
>

```

**Attributes**

The following sections describe the DVAL\_ID attributes.

**DIMENSION\_ID**

Specifies the dimension ID. This provides a way of specifying the DIMENSION\_ID as an attribute of DVAL\_ID.

**ID**

Specifies the dimension value ID.

**Sub-elements**

The DVAL\_ID element has no sub-elements.

**Example**

In this example, a DVAL\_ID is associated with a DVAL\_REF.

```

<DIMENSION_REF DIM_SEARCH_HIERARCHY="FALSE" HIDDEN="TRUE"
  RECORD_SEARCH_HIERARCHY="FALSE" TYPE="PRIMARY">
  <DIMENSION_ID ID="1" />
</DIMENSION_REF>
<DVAL_REF RANK="75">
  <DVAL_ID ID="2116" />
</DVAL_REF>
. . .

```

**DVAL\_PATH**

The DVAL\_PATH element provides an alternative way to specify a dimension value. It specifies the names of all of the dimension values in a path, starting with the root of the dimension and including the name of the target dimension value itself.

**DTD**

```

<!ELEMENT DVAL_PATH EMPTY>
<!ATTLIST DVAL_PATH
  DIMENSION_NAME  CDATA    #IMPLIED
  PATH            CDATA    #REQUIRED
>

```

**Attributes**

The following sections describe the DVAL\_PATH attributes.

**DIMENSION\_NAME**

The name of the target dimension value.

**PATH**

The full path to the target dimension value, starting with the root of the dimension. Nodes in the path are separated by a forward slash (/). If the target dval contains a forward slash, you have to add a back slash to the path (to escape the character) before adding the forward slash separator. See the example below.

### Sub-elements

The DVAL\_PATH element has no sub-elements.

### Example

This example shows two DVAL\_PATH elements declared in a precedence\_rules.xml file. The first DVAL\_PATH represents an "Endeca" DIMENSION\_NODE followed by a nested "Endeca-Cat". Neither of these two individual dimension values has a forward slash as part of their SYN PCDATA and consequently, the path can be separated with the forward slash. The second DVAL\_PATH entry represents an "A-root" DIMENSION\_NODE followed by a nested "Value/X". Because "Value/X" contains a forward slash, the PATH value requires "\" before "/X".

```
<PRECEDENCE_RULE TYPE="STANDARD">
  <DVAL_PATH PATH="Endeca/Endeca-Cat" />
  <DVAL_PATH DIMENSION_NAME="A" PATH="A-root/Value\X" />
</PRECEDENCE_RULE>
```

## KEY\_DIMENSION

The KEY\_DIMENSION element determines how records are compared by the join implementation. They are used to create a record key for a record from the specified source.

### DTD

```
<!ELEMENT KEY_DIMENSION EMPTY>
<!ATTLIST KEY_DIMENSION
  TYPE          (PVAL | DVAL | PROPERTY_VALUE)  #REQUIRED
  ID            CDATA                          #REQUIRED
  MIN_VALS     CDATA                          " 0 "
>
```

### Attributes

The following section describes KEY\_DIMENSION element's attributes.

#### TYPE

Select PVAL for property values or DVAL for dimension values. PROPERTY\_VALUE has been deprecated. Comparison of a value for a key dimension of type PVAL implies comparison of the value of the properties (the name of the property values are not considered). For key dimensions of type DVAL, the name of the dimension values are compared (neither the dimension name nor the dimension ID influence the comparison).

#### ID

The ID attribute is the property name (when TYPE=PVAL) or dimension ID (when TYPE=DVAL) of the index key.

#### MIN\_VALS

Specifies the minimum number of values.

### Sub-elements

The KEY\_DIMENSION element has no sub-elements.

## Example

The following example is taken from a typical record assembler.

```

<RECORD_JOIN JOIN_TYPE="LEFT_JOIN">
  <JOIN_ENTRY>
    <RECORD_SOURCE>rs-1</RECORD_SOURCE>
    <KEY_DIMENSION TYPE="PVAL" ID="sid"/>
  </JOIN_ENTRY>
  ...

```

## LOG

The LOG element creates a log file for Pipeline.epx components that write to a log. For example the DVAL MATCH expression often writes to a log during processing. This logging information is not the same as the output messages from Forge about record processing.



**Note:** The LOG element has been deprecated. See the *Endeca Log Server and Report Generator Guide* for details about how to implement logging in your Endeca application.

## DTD

```

<!ELEMENT LOG EMPTY>
<!ATTLIST LOG
  NAME CDATA #REQUIRED
  TYPE (SIMPLE | UNIQUE) #REQUIRED
  URL CDATA #IMPLIED
  FILE CDATA #IMPLIED
  %compression;
>

```

### Attributes

The following sections describe the LOG element's attributes.

#### NAME

The name of the log file.

#### TYPE

The type of the log file. SIMPLE indicates that the individual messages should be logged consecutively as they come in. UNIQUE indicates that each message should be made unique by removing duplicates.

#### URL

Provides a path to the log file.

#### FILE

FILE has been deprecated; use URL instead.

#### COMPRESSION\_LEVEL

Controls the level of data compression when writing log files. The default compression level for this attribute is 0 (no compression). The value "-1" for this attribute has been deprecated. Values 1-9 indicate increasing levels of compression.

**Sub-elements**

The LOG element has no sub-elements.

**Example**

This example establishes a log pipeline.

```
<LOG URL="../../../logs/pipeline.log" NAME="LogFile" TYPE="UNIQUE"/>
```

## PHRASE

The PHRASE element specifies a phrase entry for use with the Automatic Phrasing feature. The MDEX Engine treats each phrase entry as a search phrase rather than as separate search terms.

When automatic phrasing is implemented in an application, the MDEX Engine can suggest automatically phrased queries to a user or process automatically phrased queries by default. For example, if a user enters the search terms [Kenneth Cole blue jeans], and those terms are specified as in the example below, the application can return Did you mean ["Kenneth Cole" "blue jeans"]? Or depending on how automatic phrasing is implemented, the application may process the query as ["Kenneth Cole" "blue jeans"]. For details about automatic phrasing, see the *Endeca Development Guide*.

**DTD**

```
<!ELEMENT PHRASE (#PCDATA)>
```

**Attributes**

The PHRASE element has no attributes.

**Sub-elements**

The PHRASE element has no sub-elements.

**Example**

The example below adds two phrases to the phrase dictionary.

```
<PHRASE_IMPORT>
  <PHRASE>Kenneth Cole</PHRASE>
  <PHRASE>blue jeans</PHRASE>
</PHRASE_IMPORT>
```

## PROFILE

The PROFILE element specifies a user type or user group.

Profiles are used as a type of dynamic business rule trigger. See "Implementing User Profiles" in the *Endeca MDEX Engine Development Guide* for more information.

**DTD**

```
<!ELEMENT PROFILE (#PCDATA)>
```

**Attributes**

The PROFILE element has no attributes.

**Sub-elements**

The PROFILE element has no sub-elements.

**Example**

This example identifies a user type named premium\_subscriber.

```
<PROFILE>premium_subscriber</PROFILE>
```

**PROP**

The PROP element represents a property.

**DTD**

```
<!ELEMENT PROP (PVAL)>
<!ATTLIST PROP
  NAME      CDATA      #REQUIRED
  UPDATE    (A | C | D) #IMPLIED
>
```

**Attributes**

The following sections describe the PROP element's attributes.

**NAME**

Identifies the name of the property.

**Sub-elements**

The PROP element has no sub-elements.

**Example**

This example shows a property name.

```
<RECORD>
  <PROP NAME="Endeca.Title">
    <PVAL>The Simpsons Archive</PVAL>
  </PROP>
  ...
```

**PROPNAME**

The PROPNAME element represents a property.

**DTD**

```
<!ELEMENT PROPNAME (#PCDATA)>
```

**Attributes**

The PROPNAME element has no attributes.

**Sub-elements**

The PROPNAME element has no sub-elements.

**Example**

This example identifies records to update by the Number property.

```
<RECORD_SPEC>  
  <PROPNAME>Number</PROPNAME>  
</RECORD_SPEC>
```

## PVAL

The PVAL element represents a property value.

**DTD**

```
<!ELEMENT PVAL (#PCDATA)>
```

**Attributes**

The PVAL element has no attributes.

**Sub-elements**

The PVAL element has no sub-elements.

**Example**

This example shows a property value.

```
<PROP NAME="Endeca.Title">  
  <PVAL>The Simpsons Archive</PVAL>  
</PROP>
```

## RECORD\_GROUP

The RECORD\_GROUP element provides property and dimension keys for record grouping.

**DTD**

```
<ELEMENT RECORD_GROUP (KEY_DIMENSION+)>
```

**Attributes**

The RECORD\_GROUP element has no attributes.

**Sub-elements**

KEY\_DIMENSION: Determines how records are to be compared by the join implementation.

## RECORD\_ID

The RECORD\_ID element specifies the ID number for a record.

**DTD**

```
<!ELEMENT RECORD_ID EMPTY>
<!ATTLIST RECORD_ID
  ID      CDATA      #REQUIRED
>
```

**Attributes**

The following section describes the RECORD\_ID element's attribute.

**ID**

The ID attribute provides the ID number of a records.

**Sub-elements**

The RECORD\_ID element has no sub-elements.

**RECORD\_INDEX**

The RECORD\_INDEX element specifies a common key used to identify records for record retrieval, join operations, sort validation, and logging.

**DTD**

```
<!ELEMENT RECORD_INDEX (KEY_DIMENSION+)>
<!ATTLIST RECORD_INDEX
  UNIQUE      (TRUE | FALSE)      #IMPLIED
>
```

**Attributes**

The following section describes the RECORD\_INDEX element's attribute.

**UNIQUE**

Specifies whether the property or dimension key is unique or not. When set to `TRUE`, Forge checks for duplicated keys and warns if they exist. That is the attribute's behavior when it is nested in any pipeline component except a record cache. This attribute behaves differently when it is nested within a parent RECORD\_CACHE element. When set to `TRUE`, Forge discards any records with duplicate keys and logs a warning that specifies the number of records discarded.

**Sub-elements**

The following table provides a brief overview of the RECORD\_INDEX sub-elements. Click a sub-element name to view complete details.

**Example**

The example below specifies that the source contains records sorted by BID and can look up records on the same.

```
<RECORD_ADAPTER ...>
  <RECORD_INDEX>
    <KEY_DIMENSION ID="BID" TYPE="PVAL" />
  </RECORD_INDEX>
</RECORD_ADAPTER>
```

## Derived\_props.xml elements

The Derived\_props.xml file specifies derived properties for aggregated records in your pipeline.

You can specify any number of derived properties in the DERIVED\_PROPS element. See "Using Derived Properties" in the *Endeca Development Guide* for more information.

### DERIVED\_PROP

A DERIVED\_PROP element applies a function to property or dimension values from the member records of an aggregated record.

The resultant derived property is assigned to the aggregated record, giving you more control over representative values as well as over sort. See "Using Derived Properties" in the *Endeca MDEX Engine Development Guide* for more information.

#### DTD

```
<!ELEMENT DERIVED_PROP EMPTY>
<!ATTLIST DERIVED_PROP
  NAME          CDATA          #REQUIRED
  FCN           (MIN | MAX | SUM | AVG) #REQUIRED
  DERIVE_FROM   CDATA          #REQUIRED
>
```

#### Attributes

The following section describes DERIVED\_PROP element's attributes.

#### NAME

A unique name for this derived property.

#### FCN

The function you want performed on the specified dimension or property to generate the derived property. One of the following values: MIN, MAX, AVG, or SUM.

#### DERIVE\_FROM

The name of the property or dimension from which the derived property is drawn.

#### Sub-elements

DERIVED\_PROP contains no sub-elements.

#### Example

In this example, a derived property called priceDerived is generated.

```
<DERIVED_PROP FCN="SUM" NAME="priceDerived"
  DERIVE_FROM="P_PriceStr"/>
```

### DERIVED\_PROPS

A DERIVED\_PROPS element contains any number of derived properties indicated by DERIVED\_PROP elements.

**DTD**

```
<!ELEMENT DERIVED_PROPS (COMMENT?, DERIVED_PROP*)>
```

**Attributes**

The DERIVED\_PROPS element has no attributes.

**Sub-elements**

The following table provides a brief overview of the DERIVED\_PROPS sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <code>&lt;!-- ... --&gt;</code> .
DERIVED_PROP	Applies a function to property or dimension values from the member records of an aggregated record.

**Example**

This example generates two derived properties called maxPrice and minPrice.

```
<DERIVED_PROPS>
  <DERIVED_PROP FCN="MAX" NAME="maxPrice" DERIVE_FROM="p_Price"/>
  <DERIVED_PROP FCN="MIN" NAME="minPrice" DERIVE_FROM="p_Price"/>
</DERIVED_PROPS>
```

## Dimensions.xml elements

The Dimensions.xml file defines the hierarchical structure of your source data using dimensions and dimension values.

These dimensions and dimension values are what Oracle Endeca Commerce uses to classify your source data for navigation. Dimensions.xml can describe dimensions created by Developer Studio (internal, autogen or property mapped dimensions) or dimensions created using a third-party tool (external dimensions).

If it is convenient for organizational purposes, you can include multiple Dimensions.xml files in your pipeline. See the *Endeca MDEX Engine Development Guide* for more detailed conceptual information.

## BOUND

A BOUND element specifies the boundary constraints for a range of dimension values. The attributes of a BOUND element provide data type, value, and inclusion rules for the range value. Use BOUND as a sub-element of LBOUND and UBOUND.

**DTD**

```
<!ELEMENT BOUND EMPTY>
<!ATTLIST BOUND
  TYPE          (STRING | INTEGER | FLOAT | CURRENCY | DATE) #REQUIRED
  VALUE         CDATA #REQUIRED
  CLOSURE       (OPEN | CLOSED) #REQUIRED
>
```

## Attributes

The following sections describe the BOUND element's attributes.

### TYPE

The data type of the BOUND value. Must be set equal to one of the following: STRING, INTEGER, FLOAT, or CURRENCY. ( DATE is not supported. )

### VALUE

The desired value for the boundary. This value must correspond to the data type specified in TYPE.

### CLOSURE

Closure is a mathematical term that specifies whether the VALUE is included or excluded from the boundary range. CLOSED means the value is included in the range. OPEN means numbers up to or down to the VALUE are included, but the VALUE itself is not included. Must be set equal to OPEN or CLOSED.

## Sub-elements

The BOUND element has no sub-elements.

## Example

This example shows two BOUND elements that define a range of 0 to 10 for the dimension value 0 to 10. The values 0 and 10 are included in the range.

```

<DIMENSION_NODE>
  <DVAL TYPE="RANGE">
    <DVAL_ID ID="21"/>
    <SYN SEARCH="FALSE" DISPLAY="TRUE" CLASSIFY="FALSE">0 to 10</SYN>
    <LBOUND>
      <BOUND TYPE="INTEGER" VALUE="0" CLOSURE="CLOSED"/>
    </LBOUND>
    <UBOUND>
      <BOUND TYPE="INTEGER" VALUE="10" CLOSURE="CLOSED"/>
    </UBOUND>
  </DVAL>
</DIMENSION_NODE>

```

## DIMENSION

A DIMENSION element defines a single dimension for a navigation hierarchy. You can create dimensions using Developer Studio which are called internal dimensions. You can create dimensions based on external taxonomies, which are called external dimensions. The MDEX Engine requires one primary dimension, typically named Endeca. The dimensions that follow are secondary and typically reflect the logical categories of your source data.

## DTD

```

<!ELEMENT DIMENSION (DIMENSION_ID, DIMENSION_NODE)>
<!ATTLIST DIMENSION
  NAME          CDATA          #IMPLIED
  SRC_TYPE      (INTERNAL | PROPMAPPER | AUTOGEN | EXTERNAL) #IMPLIED
  SRC_FILE      CDATA          #IMPLIED
>

```

## Attributes

The following section describes the DIMENSION element's attribute.

### NAME

Specifies a unique name for the dimension.

### SRC\_TYPE

Specifies how the dimension was created and whether the dimension can be edited using Developer Studio. This attribute is for internal use by Oracle Endeca Guided Search and should not be modified. The following values are valid:

- INTERNAL Indicates a pipeline developer manually created the dimension using Developer Studio.
- PROPMAPPER Indicates Developer Studio created the dimension during property mapping. Developer Studio cannot edit the dimension.
- AUTOGEN Indicates Developer Studio automatically generated the dimension. Developer Studio cannot edit the dimension.
- EXTERNAL Indicates a third-party tool created the dimension. Developer Studio cannot edit the dimension.

### SRC\_FILE

Specifies the source file from which the dimension originated, if applicable. This attribute is for internal use by Oracle Endeca Guided Search and should not be modified.

## Sub-elements

The following table provides a brief overview of the DIMENSION sub-elements. Click a sub-element name to view complete sub-element details.

Sub-element	Brief description
DIMENSION_ID	Specifies the ID number of a dimension.
DIMENSION_NODE	Provides additional information about the DIMENSION element and allows you to nest additional DIMENSION_NODE sub-elements to create a deeper dimension hierarchy within DIMENSION.

## Example

This example shows a dimension named Year.

```
<DIMENSION NAME="Year" SRC_TYPE="INTERNAL">
  <DIMENSION_ID ID="2" />
  <DIMENSION_NODE>
    <DVAL TYPE="EXACT">
      <DVAL_ID ID="2" />
      <SYN SEARCH="FALSE" DISPLAY="TRUE" CLASSIFY="FALSE">Year</SYN>
    </DVAL>
  </DIMENSION_NODE>
</DIMENSION>
```

## DIMENSIONS

The DIMENSIONS element is the root element of the Dimension.xml file. It serves as a container for any number of DIMENSION elements which organize your navigation hierarchy. DIMENSIONS may also include the number of dimension values in the file.

**DTD**

```
<!ELEMENT DIMENSIONS (NUM_DVALS?, DIMENSION*)>
<!ATTLIST DIMENSIONS
  VERSION    CDATA    #IMPLIED
>
```

**Attributes**

The following sections describe the DIMENSIONS element's attributes.

**VERSION**

This attribute is not currently used.

**Sub-elements**

The following table provides a brief overview of the DIMENSIONS sub-elements. Click a sub-element name to view complete sub-element details.

Sub-element	Brief description
NUM_DVALS	Specifies the total number of DVAL sub-elements after Forge processes your source data.
DIMENSION	Defines a single dimension for a navigation hierarchy.

**Example**

Defines a single dimension for a navigation hierarchy.

```
<DIMENSIONS VERSION="1.0.0">
  <DIMENSION NAME="Wine Type" SRC_TYPE="INTERNAL">
    ...(any number of DIMENSION elements may appear here)...
  </DIMENSION>
</DIMENSIONS>
```

**DIMENSION\_NODE**

A DIMENSION\_NODE element serves two functions: it provides additional information about the DIMENSION element, and it allows you to nest additional DIMENSION\_NODE sub-elements to create a deeper dimension hierarchy within the parent DIMENSION.

**DTD**

```
<!ELEMENT DIMENSION_NODE (DVAL, DIMENSION_NODE*)>
```

**Attributes**

The DIMENSION\_NODE element does not have any attributes.

**Sub-elements**

The following table provides a brief overview of the DIMENSION\_NODE sub-elements. Click a sub-element name to view complete details.

Sub-elements	Brief description
DVAL	Contains an ID, one or more synonyms, and may contain boundary values for the DIMENSION.
DIMENSION_NODE	Provides additional information about the DIMENSION element and allows you to nest additional DIMENSION_NODE sub-elements to create a deeper dimension hierarchy within DIMENSION.

### Example

This example shows two DIMENSION\_NODE elements nested within the parent DIMENSION named Designation. It also shows how the two additional dimension nodes named Highly Recommended and Best Buy create a deeper hierarchy under Designation.

```
<DIMENSION NAME="Designation" SRC_TYPE="INTERNAL">
  <DIMENSION_ID ID="7"/>
  <DIMENSION_NODE>
    <DVAL TYPE="EXACT">
      <DVAL_ID ID="7"/>
      <SYN SEARCH="FALSE" DISPLAY="TRUE" CLASSIFY="FALSE">Designation</SYN>
    </DVAL>
    <DIMENSION_NODE>
      <DVAL TYPE="EXACT">
        <DVAL_ID ID="8029"/>
        <SYN SEARCH="FALSE" DISPLAY="TRUE" CLASSIFY="TRUE">Highly Recommended</SYN>
      </DVAL>
    </DIMENSION_NODE>
    <DIMENSION_NODE>
      <DVAL TYPE="EXACT">
        <DVAL_ID ID="8031"/>
        <SYN SEARCH="FALSE" DISPLAY="TRUE" CLASSIFY="TRUE">Best Buy</SYN>
      </DVAL>
    </DIMENSION_NODE>
  </DIMENSION_NODE>
</DIMENSION>
```

## DVAL

A DVAL element contains an ID, one or more synonyms, property matching instructions, and may contain boundary values for the DIMENSION.

### DTD

```
<!ELEMENT DVAL (DVAL_ID, SYN+, LBOUND?, UBOUND?, PROP*)>
<!ATTLIST DVAL
  TYPE (EXACT | RANGE | LEVEL | SIFT | DVAL) #REQUIRED
>
```

### Attributes

The following sections describe the DVAL element's attributes.

#### TYPE

The type attribute specifies how property values are matched to synonyms during classification. Any of the following values are acceptable:

- EXACT — Matches if the text within SYN is an exact match. For example, "Year" and "Year" are an exact match; however, "Year" and "YEAR" are not.
- RANGE — Matches if the property value falls within the specified range. See the BOUND element for more information about specifying a range of values.
- LEVEL — Matches different properties at different levels in the hierarchy. Note that LEVEL is no longer supported.
- SIFT — Matches if the property value falls within a sifted range several layers within a dimension hierarchy. Use the BOUND element to specify the range to match.
- DVAL — DVAL is deprecated. Use EXACT instead.

### Sub-elements

The following table provides a brief overview of the DVAL sub-elements.

Sub-element	Brief description
DVAL_ID	Specifies the dimension value ID.
SYN	Specifies synonyms and processing settings for the particular DIMENSION value.
LBOUND	Specifies the lower boundary constraints for a range of dimension values.
UBOUND	Specifies the upper boundary constraints for a range of dimension values.
PROP	Represents a property.

### Example

This example defines a dimension value with the following characteristics:

- An ID of 30
- A displayable synonym of 91 to 100
- A range of 91 to 100

```
<DIMENSION_NODE>
  <DVAL TYPE="RANGE">
    <DVAL_ID ID="30"/>
    <SYN SEARCH="FALSE" DISPLAY="TRUE" CLASSIFY="FALSE">91 to 100</SYN>
    <LBOUND>
      <BOUND TYPE="INTEGER" VALUE="91" CLOSURE="CLOSED"/>
    </LBOUND>
    <UBOUND>
      <BOUND TYPE="INTEGER" VALUE="100" CLOSURE="CLOSED"/>
    </UBOUND>
  </DVAL>
</DIMENSION_NODE>
```

## LBOUND

A LBOUND element specifies the lower boundary constraints for a range dimension value. Both LBOUND and UBOUND require the sub-element BOUND to provide data type, value, and inclusion rules for the range value.

**DTD**

```
<!ELEMENT LBOUND (BOUND)>
```

**Attributes**

The LBOUND element does not have any attributes.

**Sub-elements**

The following table provides a brief overview of the LBOUND sub-elements.

Sub-element	Brief description
BOUND	Specifies the boundary constraints for a range dimension value.

**Example**

This example shows the LBOUND element defining a lower value of 0 in range of 0 to 10. The UBOUND element defines the upper range of 10. The values 0 and 10 are included in the range.

```
<DIMENSION_NODE>
  <DVAL TYPE="RANGE">
    <DVAL_ID ID="21" />
    <SYN SEARCH="FALSE" DISPLAY="TRUE" CLASSIFY="FALSE">0 to 10</SYN>
    <LBOUND>
      <BOUND TYPE="INTEGER" VALUE="0" CLOSURE="CLOSED" />
    </LBOUND>
    <UBOUND>
      <BOUND TYPE="INTEGER" VALUE="10" CLOSURE="CLOSED" />
    </UBOUND>
  </DVAL>
</DIMENSION_NODE>
```

**NUM\_DVALS**

A NUM\_DVALS element specifies the total number of DVAL sub-elements after Forge processes your source data. This element does not appear in the dimension.xml files used as input to Forge. This is not an element that you should add to your dimension.xml input files. Forge calculates this value and writes it to dimensions.xml. Dgidx uses this value; you should not edit it.

**DTD**

```
<!ELEMENT NUM_DVALS (#PCDATA)>
```

**Attributes**

The NUM\_DVALS element does not have any attributes.

**Sub-elements**

The NUM\_DVALS element does not have any sub-elements.

## Example

This example shows the number of DVALS that appear after Forge processes the source data.

```
<DIMENSIONS VERSION="1.0.0">
  <NUM_DVALS>8067</NUM_DVALS>
```

## SYN

A SYN element specifies synonyms and processing settings for a particular DIMENSION value.

Synonyms provide alternative ways of describing and consequently, searching a particular dimension. If you created your dimension elements within the Endeca Developer Studio, Developer Studio generates XML with the same name for SYN and DIMENSION value.

## DTD

```
<!ELEMENT SYN (#PCDATA)>
<!ATTLIST SYN
  DISPLAY      (TRUE | FALSE)    #REQUIRED
  SEARCH       (TRUE | FALSE)    #REQUIRED
  CLASSIFY     (TRUE | FALSE)    #REQUIRED
>
```

## Attributes

The following sections describe the SYN element's attributes.

### DISPLAY

Specifies which synonym is displayed to the user. TRUE indicates the synonym is displayed. FALSE indicates it is not.

### SEARCH

Specifies whether Dgidx indexes the dimension. TRUE indicates that the value is indexed during property-to-dimension mapping. FALSE indicates it is not.

### CLASSIFY

Controls property-to-dimension mapping. TRUE indicates that the value is used during property-to-dimension mapping. FALSE indicates the value is not.

## Sub-elements

The SYN element does not have any sub-elements.

## Example

This example shows SYN elements where US, USA, and United States are synonyms but only "United States" displays for dimension searches using "US" or "USA".

```
<DIMENSION NAME="Country" SRC_TYPE="INTERNAL">
  <DIMENSION_ID ID="2"/>
  <DIMENSION_NODE>
    <DVAL TYPE="EXACT">
      <DVAL_ID ID="2"/>
      <SYN SEARCH="TRUE" DISPLAY="FALSE" CLASSIFY="TRUE">USA</SYN>
      <SYN SEARCH="TRUE" DISPLAY="FALSE" CLASSIFY="FALSE">US</SYN>
      <SYN SEARCH="FALSE" DISPLAY="TRUE" CLASSIFY="TRUE">United States</SYN>
    </DVAL>
  </DIMENSION_NODE>
</DIMENSION>
```

```
</DIMENSION_NODE>
</DIMENSION>
```

## UBOUND

A UBOUND element specifies the upper boundary constraints for a range of dimension values. Both UBOUND and LBOUND require the sub-element BOUND to provide data type, value, and inclusion rules for the range value.

### DTD

```
<!ELEMENT UBOUND (BOUND)>
```

### Attributes

The UBOUND element does not have any attributes.

### Sub-elements

The following table provides a brief overview of the UBOUND sub-elements.

Sub-element	Brief description
BOUND	Specifies the boundary constraints for a range dimension value.

### Example

This example shows the LBOUND element defining a lower value of 0 in range of 0 to 10. The UBOUND element defines the upper range of 10. The values 0 and 10 are included in the range.

```
<DIMENSION_NODE>
  <DVAL TYPE="RANGE">
    <DVAL_ID ID="21" />
    <SYN SEARCH="FALSE" DISPLAY="TRUE" CLASSIFY="FALSE">0 to 10</SYN>
    <LBOUND>
      <BOUND TYPE="INTEGER" VALUE="0" CLOSURE="CLOSED" />
    </LBOUND>
    <UBOUND>
      <BOUND TYPE="INTEGER" VALUE="10" CLOSURE="CLOSED" />
    </UBOUND>
  </DVAL>
</DIMENSION_NODE>
```

## Dimension\_groups.xml elements

The Dimension\_groups.xml file allows you to organize dimensions into explicit groupings for presentation purposes.

Explicit dimension groups that you establish provide a way to impose relationships on dimensions and take precedence over implicit, system-generated dimension groups containing a single member each.

A dimension can only belong to one dimension group. For example, if a data set has ten dimensions and one explicit dimension group containing five dimensions has been created, then there are six dimension groups: one explicit group and five implicit ones.

## DIMENSION\_GROUP

The DIMENSION\_GROUP element specifies that a group of dimensions should be returned together when sent back from the MDEX Engine.

The order of the dimension groups is determined by the highest ranked dimension within each group. To control the order in which dimensions are returned, you should use the DVAL\_REF element to specify a RANK for the root dimension value of the dimension. That same rank determines how dimensions are ordered within a dimension group.

### DTD

```
<!ELEMENT DIMENSION_GROUP (DIMNAME+)>
<!ATTLIST DIMENSION_GROUP
  NAME      CDATA      #REQUIRED
>
```

### Attributes

The following section describes the DIMENSION\_GROUP element's attribute.

#### NAME

A unique name for this dimension group.

### Sub-elements

The following table provides a brief overview of the DIMENSION\_GROUP sub-elements.

Sub-element	Brief description
DIMNAME	Specifies the name of a dimension. This element provides an alternative to identifying a dimension by its ID with DIMENSION_ID.

### Example

This example shows dimension groups used in the wine reference implementation.

```
<DIMENSION_GROUPS>
  <DIMENSION_GROUP NAME="Characteristics">
    <DIMNAME>Body</DIMNAME>
  </DIMENSION_GROUP>
  <DIMENSION_GROUP NAME="Ratings">
    <DIMNAME>Drinkability</DIMNAME>
    <DIMNAME>Designation</DIMNAME>
    <DIMNAME>Score</DIMNAME>
  </DIMENSION_GROUP>
  <DIMENSION_GROUP NAME="Geographic Information">
    <DIMNAME>Appellations</DIMNAME>
    <DIMNAME>Region</DIMNAME>
    <DIMNAME>Winery</DIMNAME>
  </DIMENSION_GROUP>
</DIMENSION_GROUPS>
```

## DIMENSION\_GROUPS

A DIMENSION\_GROUPS element contains any number of dimension groups indicated by DIMENSION\_GROUP elements.

**DTD**

```
<!ELEMENT DIMENSION_GROUPS (COMMENT?, DIMENSION_GROUP*)>
```

**Attributes**

The DIMENSION\_GROUPS element has no attributes.

**Sub-elements**

The following table provides a brief overview of the DIMENSION\_GROUP sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.
DIMENSION_GROUP	Specifies that a group of dimensions should be returned together when sent back from the MDEX Engine.

**Example**

This example shows dimension groups used in the wine reference implementation.

```
<DIMENSION_GROUPS>
  <DIMENSION_GROUP NAME="Characteristics">
    <DIMNAME>Body</DIMNAME>
  </DIMENSION_GROUP>
  <DIMENSION_GROUP NAME="Ratings">
    <DIMNAME>Drinkability</DIMNAME>
    <DIMNAME>Designation</DIMNAME>
    <DIMNAME>Score</DIMNAME>
  </DIMENSION_GROUP>
  <DIMENSION_GROUP NAME="Geographic Information">
    <DIMNAME>Appellations</DIMNAME>
    <DIMNAME>Region</DIMNAME>
    <DIMNAME>Winery</DIMNAME>
  </DIMENSION_GROUP>
</DIMENSION_GROUPS>
```

## Dimension\_refs.xml elements

The Dimension\_refs.xml file contains a DIMENSION\_REF element for every dimension in your hierarchy.

The DIMENSION\_REF allows you to specify whether a dimension is hidden in the application's user interface and how dimension values may be returned when more than one dimension value is selected from a dimension. DIMENSION\_REF elements are grouped in the DIMENSION\_REFS element.

### DIMENSION\_REF

The DIMENSION\_REF element describes the various attributes of a dimension.

**DTD**

```
<!ELEMENT DIMENSION_REF EMPTY>
<!ATTLIST DIMENSION_REF
  NAME          CDATA          #REQUIRED
  TYPE          (PRIMARY | SECONDARY) #REQUIRED
  HIDDEN        (TRUE | FALSE)    "FALSE"
  MULTI        (OR | AND)        #IMPLIED
>
```

**Attributes**

The following sections describe the DIMENSION\_REF element's attributes.

**NAME**

Specifies a unique name for the DIMENSION\_REF.

**TYPE**

In pre-6.1 implementations, this attribute specified whether this dimension was a primary dimension or a secondary dimension. As of version 6.1, this distinction is now ignored and all dimensions are treated the same. That is, regardless of whether a dimension is configured as type PRIMARY or SECONDARY, the value is ignored by Dgidx and the MDEX Engine. However, because the TYPE attribute is required by Developer Studio, it is recommended that you specify SECONDARY as the type for all dimensions.

**HIDDEN**

Specifies whether or not this dimension is shown in the navigation controls. The default value is FALSE.

**MULTI**

Allows the user to select more than one dimension value from a dimension. When set to OR, the MDEX Engine returns dimension values that match either search term. When set to AND, the MDEX Engine returns dimension values that match both search terms.

**Sub-elements**

DIMENSION\_REF contains no sub-elements.

**Example**

This example shows a small number of dimensions in a pipeline that may appear in the application's navigation controls.

```
<DIMENSION_REFS>
  <DIMENSION_REF NAME="Camcorder Features" TYPE="SECONDARY" HIDDEN="FALSE"/>
  <DIMENSION_REF NAME="Camcorder Formats" TYPE="SECONDARY" HIDDEN="FALSE"/>
  <DIMENSION_REF NAME="Camcorder Size" TYPE="SECONDARY" HIDDEN="FALSE"/>
  <DIMENSION_REF NAME="Camera Types" TYPE="SECONDARY" HIDDEN="FALSE"/>
  ...Others deleted for simplicity...
</DIMENSION_REFS>
```

**DIMENSION\_REFS**

A DIMENSION\_REFS element contains any number of dimensions indicated by DIMENSION\_REF elements.

**DTD**

```
<!ELEMENT DIMENSION_REFS (COMMENT?, DIMENSION_REF*)>
```

## Attributes

The DIMENSION\_REFS element has no attributes.

## Sub-elements

The following table provides a brief overview of the DIMENSION\_REFS sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.
DIMENSION_REFS	Describes the various attributes of a dimension.

## Example

This example shows a small number of dimensions in a pipeline that may appear in the application's navigation controls.

```
<DIMENSION_REFS>
  <DIMENSION_REF NAME="Camcorder Features" TYPE="SECONDARY" HIDDEN="FALSE" />
  <DIMENSION_REF NAME="Camcorder Formats" TYPE="SECONDARY" HIDDEN="FALSE" />
  <DIMENSION_REF NAME="Camcorder Size" TYPE="SECONDARY" HIDDEN="FALSE" />
  <DIMENSION_REF NAME="Camera Types" TYPE="SECONDARY" HIDDEN="FALSE" />
  ...Others deleted for simplicity...
</DIMENSION_REFS>
```

# Dimsearch\_index.xml elements

The Dimsearch\_index.xml file specifies how Dgidx should index dimensions for dimension searches.

If you modify this file, you must re-index your data for the changes to take effect. Dgidx should index dimensions for dimension searches.

## DIMSEARCH\_HIERARCHY

A DIMSEARCH\_HIERARCHY element returns the specified dimension values when a search matches multiple dimension values in a data hierarchy.

### DTD

```
<!ELEMENT DIMSEARCH_HIERARCHY EMPTY>
<!ATTLIST DIMSEARCH_HIERARCHY
  NAME      CDATA      #REQUIRED
>
```

### Attributes

The following sections describe the DIMSEARCH\_HIERARCHY element's attribute.

#### NAME

Specifies the name of a dimension.

## Sub-elements

DIMSEARCH\_HIERARCHY contains no sub-elements.

## Example

```
<DIMSEARCH_INDEX COMPOUND="FALSE" DICTIONARY_MAX_NGRAM_LENGTH="5" DICTIONARY_WILDCARD="TRUE"
INMEM_INDEX_THRESHOLD="512" MAX_NGRAM_LENGTH="3" MAX_WORD_LENGTH="16"
MIN_WORD_LENGTH="3"
MIN_WORD_OCCURRENCES="1">
  <SEARCH_INDEX/>
  <DIMSEARCH_HIERARCHY NAME="Review Score"/>
</DIMSEARCH_INDEX>
```

## DIMSEARCH\_INDEX

A DIMSEARCH\_INDEX element controls the behavior of search on a dimension.

### DTD



**Note:** Several settings in this element are ignored starting with the MDEX Engine version 6.1.2. These settings are: MAX\_NGRAM\_LENGTH, DICTIONARY\_MAX\_NGRAM\_LENGTH, and DICTIONARY\_WILDCARD. Do not remove these settings from the file, since the DTD depends on them. Starting with the version 6.1.2, the MDEX Engine uses a simplified mechanism for wildcard search. This mechanism does not require specifying these settings. For more information on wildcard search, see the *Endeca MDEX Engine Development Guide* version 6.1.2 or higher.

```
<!ELEMENT DIMSEARCH_INDEX (COMMENT?, SEARCH_INDEX, DIMSEARCH_HIERARCHY*)>
<!ATTLIST DIMSEARCH_INDEX
COMPOUND          (TRUE|FALSE)      "FALSE"
INMEM_INDEX_THRESHOLD CDATA      #IMPLIED
MAX_NGRAM_LENGTH  CDATA      #IMPLIED
DICTIONARY_WILDCARD (TRUE|FALSE) #IMPLIED
DICTIONARY_MAX_NGRAM_LENGTH CDATA #IMPLIED
MIN_WORD_OCCURRENCES CDATA #IMPLIED
MIN_WORD_LENGTH   CDATA #IMPLIED
MAX_WORD_LENGTH   CDATA #IMPLIED
>
```

### Attributes

The following sections describe the attributes of the DIMSEARCH\_INDEX element.

#### COMPOUND

Specifies whether the dimension is enabled for compound dimension searches. This extends a default dimension search which returns single dimensions. The default value is FALSE.

#### INMEM\_INDEX\_THRESHOLD

*In MDEX Engine 6.4.0, this attribute was deprecated.* Specifies the maximum text size (in MB) for which text indexing will be done entirely in memory. If the text size is larger than this threshold, then temporary files on disk will be used. Typically, you do not need to modify the value of this attribute. However, there are two infrequent scenarios in which you may want to adjust the value of this attribute:

- If the machine running Dgidx has less than the recommended amount of RAM, you might reduce this value.

- If other indexing structures cause Dgidx to run out of memory, you might reduce this value to allow more memory for other purposes.

### MAX\_NGRAM\_LENGTH



**Note:** The value specified in this setting is ignored in the MDEX Engine starting with its version 6.1.2. However, do not remove this XML setting.

Specifies the maximum ngram length that should be indexed. The default value is 3. All substrings of this length or shorter will be indexed. If a user does a wildcard search with a string that is less than or equal to this length, then exact results can be returned directly from the index. If the wildcard search includes a substring longer than this length, then the results returned from the index will be post-processed so that false positives are eliminated.

### DICTIONARY\_WILDCARD



**Note:** The value specified in this setting is ignored in the MDEX Engine starting with its version 6.1.2. However, do not remove this XML setting.

Specifies whether a wildcard index should be generated for the dictionary of words for the text collection containing this element. The default value of this attribute is TRUE when the containing element consists of off-line documents and FALSE otherwise.

### DICTIONARY\_MAX\_NGRAM\_LENGTH



**Note:** The value specified in this setting is ignored in the MDEX Engine starting with its version 6.1.2. However, do not remove this XML setting.

Specifies the maximum ngram length that should be indexed in the dictionary wildcard index. The default value is 5.

### MIN\_WORD\_OCCURRENCES

Specifies the minimum number of times a word must occur for it to be indexed for spelling correction. The default value is 1.

### MIN\_WORD\_LENGTH

Specifies the minimum number of a characters that a word must contain for it to be indexed for spelling correction. The default value is 3 (words that are three characters or larger are added to the index).

### MAX\_WORD\_LENGTH

Specifies the maximum number of a characters that a word may contain for it to be indexed for spelling correction. The default value is 16 (words that are 16 characters or fewer are added to the index).

### Sub-elements

The following table provides a brief overview of the DIMSEARCH\_INDEX sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.
SEARCH_INDEX	Controls the construction of the search index for dimension search.

Sub-element	Brief description
DIMSEARCH_HIERARCHY	Returns the specified dimension values when a search matches multiple dimension values in a data hierarchy.

**Example**

```
<DIMSEARCH_INDEX COMPOUND="FALSE" DICTIONARY_MAX_NGRAM_LENGTH="5" DICTIONARY_WILD-
CARD="TRUE"
INMEM_INDEX_THRESHOLD="512" MAX_NGRAM_LENGTH="3" MAX_WORD_LENGTH="16"
MIN_WORD_LENGTH="3"
MIN_WORD_OCCURRENCES="1">
  <SEARCH_INDEX/>
</DIMSEARCH_INDEX>
```

## Dval\_ranks.xml elements

The Dval\_ranks.xml file assigns static ranking to dimension values.

Static ranking defines the order in which dimensions and dimension values appear in the user interface. Each dimension's rank is specified in a DVAL\_RANK element. DVAL\_RANK elements are grouped in the DVAL\_RANKS root element. For more information, see "Using Relevance Ranking" in the *Endeca MDEX Engine Development Guide*.

### DVAL\_RANK

A DVAL\_RANK element specifies a rank value for the root dimension value of a dimension. That same rank value also determines how dimensions are ordered within a dimension group. The order of the dimension groups is determined by the highest ranked dimension within each group.

**DTD**

```
<!ELEMENT DVAL_RANK EMPTY>
<!ATTLIST DVAL_RANK
  ID          CDATA          #REQUIRED
  RANK       CDATA          #REQUIRED
>
```

**Attributes**

The following sections describe the DVAL\_RANK element's attribute.

**ID**

Specifies the dimension value ID.

**RANK**

Specifies the rank of this dimension value. This integer value determines how dimension values should be ranked when returned as refinements.

**Sub-elements**

DVAL\_RANK contains no sub-elements.

## Example

This example shows how dimension values are ranked in the wine reference implementation.

```

<DVAL_RANKS>
  <DVAL_RANK ID="30" RANK="1" />
  <DVAL_RANK ID="28" RANK="3" />
  <DVAL_RANK ID="27" RANK="4" />
  <DVAL_RANK ID="29" RANK="2" />
  <DVAL_RANK ID="26" RANK="5" />
  <DVAL_RANK ID="25" RANK="6" />
  <DVAL_RANK ID="24" RANK="7" />
  <DVAL_RANK ID="23" RANK="8" />
  <DVAL_RANK ID="22" RANK="9" />
  <DVAL_RANK ID="21" RANK="11" />
  <DVAL_RANK ID="8000" RANK="10" />
  <DVAL_RANK ID="10" RANK="9" />
  <DVAL_RANK ID="2" RANK="8" />
</DVAL_RANKS>

```

## DVAL\_RANKS

A DVAL\_RANKS element contains any number of dimension value rankings indicated by DVAL\_RANK elements.

### DTD

```

<!ELEMENT DVAL_RANKS (COMMENT?, DVAL_RANK*)>

```

### Attributes

The DVAL\_RANKS element has no attributes.

### Sub-elements

The following table provides a brief overview of the DVAL\_RANKS sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.
DVAL_RANK	Specifies a rank value for the root dimension value of a dimension.

## Example

This example shows how dimension values are ranked in the wine reference implementation.

```

<DVAL_RANKS>
  <DVAL_RANK ID="30" RANK="1" />
  <DVAL_RANK ID="28" RANK="3" />
  <DVAL_RANK ID="27" RANK="4" />
  <DVAL_RANK ID="29" RANK="2" />
  <DVAL_RANK ID="26" RANK="5" />
  <DVAL_RANK ID="25" RANK="6" />
  <DVAL_RANK ID="24" RANK="7" />
  <DVAL_RANK ID="23" RANK="8" />
  <DVAL_RANK ID="22" RANK="9" />
  <DVAL_RANK ID="21" RANK="11" />

```

```

<DVAL_RANK ID="8000" RANK="10" />
<DVAL_RANK ID="10" RANK="9" />
<DVAL_RANK ID="2" RANK="8" />
</DVAL_RANKS>

```

## Dval\_refs.xml elements

The Dval\_refs.xml file contains a DVAL\_REF element for every dimension value in your hierarchy.

The DVAL\_REF element allows you to specify a dimension value's ID, whether it is navigable or not, and whether it is collapsible or not. DVAL\_REF elements are grouped in the DVAL\_REFS parent element.

### DVAL\_REF

The DVAL\_REF element specifies how individual dimension values behave.

#### DTD

```

<!ELEMENT DVAL_REF EMPTY>
<!ATTLIST DVAL_REF
  ID              CDATA              #REQUIRED
  INERT           (TRUE | FALSE)    "FALSE"
  COLLAPSIBLE    (TRUE | FALSE)    "FALSE"
>

```

#### Attributes

The following sections describe the DVAL\_REF attributes.

##### ID

Specifies the dimension value ID.

##### INERT

Specifies whether this dimension value is navigable. TRUE indicates the dimension value is not navigable. The default value is FALSE.

##### COLLAPSIBLE

COLLAPSIBLE is used in combination with the DVAL\_COLLAPSE\_THRESHOLD attribute on REFINEMENTS. Marking an internal dimension value as collapsible means that if the navigation state returns more child dimension values than of value specified in DVAL\_COLLAPSE\_THRESHOLD of REFINEMENTS, then collapsible dimension value is returned instead of the child dimension values.

Note that the DVAL\_COLLAPSE\_THRESHOLD, which is set at the dimension level, applies to all dimension values in that dimension. Each dimension value within a dimension may be marked as COLLAPSIBLE or not. See the DVAL\_COLLAPSE\_THRESHOLD description of REFINEMENTS for an example.

#### Sub-elements

DVAL\_REF contains no sub-elements.

## Example

This example shows a number of DVAL\_REF elements that are collapsible.

```

<DVAL_REFS>
  <DVAL_REF ID="900024" COLLAPSIBLE="TRUE" />
  <DVAL_REF ID="900015" COLLAPSIBLE="TRUE" />
  <DVAL_REF ID="900006" COLLAPSIBLE="TRUE" />
  <DVAL_REF ID="900021" COLLAPSIBLE="TRUE" />
  <DVAL_REF ID="900009" COLLAPSIBLE="TRUE" />
  <DVAL_REF ID="900016" COLLAPSIBLE="TRUE" />
  <DVAL_REF ID="900012" COLLAPSIBLE="TRUE" />
  <DVAL_REF ID="900002" COLLAPSIBLE="TRUE" />
  <DVAL_REF ID="900007" COLLAPSIBLE="TRUE" />
</DVAL_REFS>

```

## DVAL\_REFS

A DVAL\_REFS element contains any number of dimension value references indicated by DVAL\_REF elements.

### DTD

```

<!ELEMENT DVAL_REFS (COMMENT?, DVAL_REF*)>

```

### Attributes

The DVAL\_REFS element has no attributes.

### Sub-elements

The following table provides a brief overview of the DVAL\_REFS sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.
DVAL_REF	Specifies how individual dimension values behave.

### Example

This example shows a number of DVAL\_REF elements contained in a parent DVAL\_REFS element.

```

<DVAL_REFS>
  <DVAL_REF ID="900024" COLLAPSIBLE="TRUE" />
  <DVAL_REF ID="900015" COLLAPSIBLE="TRUE" />
  <DVAL_REF ID="900006" COLLAPSIBLE="TRUE" />
  <DVAL_REF ID="900021" COLLAPSIBLE="TRUE" />
  <DVAL_REF ID="900009" COLLAPSIBLE="TRUE" />
  <DVAL_REF ID="900016" COLLAPSIBLE="TRUE" />
  <DVAL_REF ID="900012" COLLAPSIBLE="TRUE" />
  <DVAL_REF ID="900002" COLLAPSIBLE="TRUE" />

```

```
<DVAL_REF ID="900007" COLLAPSIBLE="TRUE" />
</DVAL_REFS>
```

## Externaldimension elements

The externaldimensions.xml file provides a means to create dimension hierarchies outside of Developer Studio. You can then import an externally created dimension into your project and use it in your Endeca application. The XML elements in this file are documented in the "Working with Externally Created Dimensions" chapter of the *Endeca Forge Guide*.

## Key\_props.xml elements

The key\_props.xml file allows property and dimension keys to be annotated with metadata key/value pairs called key properties (since they are properties of a dimension or property key).

These key properties are configured as PROP elements as part of the application configuration. The XML elements in this file are documented in the "Configuring Key Properties" chapter of the *Endeca MDEX Engine Analytics Guide*.

## Languages.xml elements

The Languages.xml file specifies language identification codes for values in named dimensions or properties.

This identification allows a single MDEX Engine to contain records in multiple languages. The root element of Languages.xml is LANGUAGES. Within LANGUAGES, you specify associations between a dimension or property and a language using KEY\_LANGUAGE. For more information, see "Using Internationalized Data" in the *Endeca Development Guide*.

## KEY\_LANGUAGE

The KEY\_LANGUAGE element instructs the MDEX Engine to treat values from a specified dimension or property as values from a specified language.

The NAME attribute specifies the dimension or property. The LANGUAGE attribute specifies the code for the language. This association is useful when Endeca records in a number of languages are handled by a single dgraph instance.

### DTD

```
<!ELEMENT KEY_LANGUAGE EMPTY>
<!ATTLIST KEY_LANGUAGE
  NAME          CDATA          #REQUIRED
  LANGUAGE      CDATA          #REQUIRED
>
```

### Attributes

The following section describes the KEY\_LANGUAGE element's attributes.

**NAME**

Specifies either a dimension or a property name to associate with a LANGUAGE value.

**LANGUAGE**

Specifies the language code for values represented in the dimension or property. ISO 639 lists the valid language codes. Common examples include the following: `en` for English, `es` for Spanish, `fr` for French, `de` for German, `ja` for Japanese, and `ko` for Korean. Chinese has two available codes: `zh-CN` for simplified Chinese and `zh-TW` for traditional Chinese.



**Note:** Language codes are case-sensitive and should be specified in lower case with the exception of `zh-TW` and `zh-CN`.

**Sub-elements**

The `KEY_LANGUAGE` element has no sub-elements.

**Example**

This example instructs the MDEX Engine to treat values in dimension called "Title\_English" as English language values, and treat values in dimension called "Title\_Spanish" as Spanish language values.

```
<LANGUAGES>
  <KEY_LANGUAGE NAME="Title_English" LANGUAGE="en" />
  <KEY_LANGUAGE NAME="Title_Spanish" LANGUAGE="es" />
</LANGUAGES>
```

**LANGUAGES**

The `LANGUAGES` element lists all of the language associations between a dimension or property and a particular language.

**DTD**

```
<!ELEMENT LANGUAGES (COMMENT?, KEY_LANGUAGE*)>
```

**Attributes**

The `LANGUAGES` element has no attributes.

**Sub-elements**

The following table provides a brief overview of the `LANGUAGES` sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <code>&lt;!-- ... --&gt;</code> .
KEY_LANGUAGE	Instructs the MDEX Engine to treat values from a specified dimension or property as values from a specified language.

### Example

This example instructs the MDEX Engine to treat values in dimension called "Title\_English" as English language values and treat values in dimension called "Title\_Spanish" as Spanish language values.

```
<LANGUAGES>
  <KEY_LANGUAGE NAME="Title_English" LANGUAGE="en" />
  <KEY_LANGUAGE NAME="Title_Spanish" LANGUAGE="es" />
</LANGUAGES>
```

## Phrases.xml elements

The Phrases.xml file specifies the recipe for building the phrase dictionary that the MDEX Engine uses for the Automatic Phrasing feature.

The root element of Phrases.xml is PHRASES. Its sub-elements (PHRASE\_ADDITIONS and DIMENSION\_IMPORTS) allow you to specify phrase entries for use with the Automatic Phrasing feature. When automatic phrasing is implemented in an application, the MDEX Engine can suggest automatically phrased queries to a user or process automatically phrased queries by default. For more information, see the "Using Automatic Phrasing" chapter in the *Endeca MDEX Engine Development Guide*.

## PHRASES

A PHRASES element specifies phrase entries for the phrase dictionary.

You use the Automatic Phrasing editor in Developer Studio to import phrases from an XML phrase file or extract phrases from dimensions.

### DTD

```
<!ELEMENT PHRASES
  ( COMMENT?
  , PHRASE_ADDITIONS
  , DIMENSION_IMPORTS
  )
>
```

### Attributes

The PHRASES element has no attributes.

### Sub-elements

The following table provides a brief overview of the PHRASES sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.
PHRASE_ADDITIONS	Represents phrases that have been imported from an XML phrase file.
DIMENSION_IMPORTS	Represents dimension names from which phrases will be extracted from their dimension values.

## Example

This example shows two phrases ("Napa Valley" and "low tannin") that were added from a phrase file and one dimension ("Wine\_Type") from which phrases will be extracted.

```
<PHRASES>
  <PHRASE_ADDITIONS>
    <PHRASE>Napa Valley</PHRASE>
    <PHRASE>low tannin</PHRASE>
  </PHRASE_ADDITIONS>
  <DIMENSION_IMPORTS>
    <DIMNAME>Wine_Type</DIMNAME>
  </DIMENSION_IMPORTS>
</PHRASES>
```

## PHRASE\_ADDITIONS

A PHRASE\_ADDITIONS element specifies a phrase entry that was imported from an XML phrase file.

You can import phrases from an XML phrase file by using the Import Phrases dialog box in Developer Studio. The phrases are added to the phrase dictionary that the MDEX Engine uses for the Automatic Phrasing feature.

### DTD

```
<!ELEMENT PHRASE_ADDITIONS (PHRASE*)>
```

### Attributes

The PHRASE\_ADDITIONS element has no attributes.

### Sub-element

The following table provides a brief overview of the PHRASE\_ADDITIONS sub-element.

Sub-element	Brief description
PHRASE	Specifies a phrase entry for the phrase dictionary. For details, see the PHRASE element which is a common element in the Common.dtd elements file.

### Example

This example shows two phrases ("Napa Valley" and "low tannin") that were added from a phrase file.

```
<PHRASES>
  <PHRASE_ADDITIONS>
    <PHRASE>Napa Valley</PHRASE>
    <PHRASE>low tannin</PHRASE>
  </PHRASE_ADDITIONS>
  <DIMENSION_IMPORTS/>
</PHRASES>
```

## DIMENSION\_IMPORTS

A DIMENSION\_IMPORTS element specifies dimension names from which phrases will be extracted from their dimension values.

The MDEX Engine adds each multi-term dimension value in a selected dimension to the phrase dictionary. Single-term dimension values are not included. For example, if you import a WineType dimension from a wine

catalog, the MDEX Engine creates a phrase entry for multi-term names such as "Pinot Noir" but not for single-term names such as "Merlot".

### DTD

```
<!ELEMENT DIMENSION_IMPORTS (DIMNAME*)>
```

### Attributes

The DIMENSION\_IMPORTS element has no attributes.

### Sub-element

The following table provides a brief overview of the DIMENSION\_IMPORTS sub-element.

Sub-element	Brief description
DIMNAME	Specifies a dimension name from which a phrase will be extracted for the phrase dictionary. For details, see the DIMNAME element which is a common element in the Common.dtd elements file.

### Example

This example shows two dimensions ("Flavors" and "Wine\_Type") from which phrases will be extracted.

```
<PHRASES>
  <PHRASE_ADDITIONS/>
  <DIMENSION_IMPORTS>
    <DIMNAME>Flavors</DIMNAME>
    <DIMNAME>Wine_Type</DIMNAME>
  </DIMENSION_IMPORTS>
</PHRASES>
```

## Pipeline.epx elements

The Pipeline.epx file defines the entire data transformation process of your source data into navigable Endeca records.

Pipeline.epx describes file locations, file formats, any necessary data manipulation, property-to-dimension mapping, and many other components in the process. Using simple NAME attributes as cross-references, these components are linked together in a flow that suggests a "pipeline" feel.

Each element available for use in Pipeline.epx is fully described in its own help topic within this section. See "Pipeline Overview" in the *Endeca Forge Guide* for more detailed conceptual information.

## DIMENSION\_ADAPTER

A DIMENSION\_ADAPTER element reads and writes dimension files in .xml or binary formats.

When the optional TRANSFORMER sub-element is included, a dimension adapter can read in .xml files from external taxonomies and from Stratify Taxonomy Manager.

### DTD

```
<!ELEMENT DIMENSION_ADAPTER
  ( COMMENT? )
```

```

    , DIMENSION_SOURCE?
    , TRANSFORMER?
    , PASS_THROUGH*
  )
>
<!ATTLIST DIMENSION_ADAPTER
  NAME                CDATA                #REQUIRED
  DIRECTION            ( INPUT | OUTPUT ) #REQUIRED
  URL                  CDATA                #REQUIRED
  COL_DELIMITER        CDATA                #IMPLIED
  ROW_DELIMITER        CDATA                #IMPLIED
  REC_DELIMITER        CDATA                #IMPLIED
  FORMAT               CDATA                #REQUIRED
  REQUIRE_DATA         ( TRUE | FALSE )    #IMPLIED
  %compression;
>

```

### Attributes

The following sections describe the DIMENSION\_ADAPTER element's attributes.

#### NAME

Identifies the DIMENSION\_ADAPTER so that it can be referenced by other components in Pipeline.epx.

#### DIRECTION

Indicates either INPUT or OUTPUT depending on whether the adapter is reading dimensions (input) or writing dimensions (output).

#### URL

Specifies the location of the source data, in this case dimension files. The path is either an absolute path or a path relative to the location of the Pipeline.epx file. With an absolute path, the protocol must be specified in RFC 2396 syntax. Usually this means the prefix file:/// precedes the path to the data file. Relative URLs must not specify the protocol and are relative to the URL used to locate the Pipeline.epx file.

#### COL\_DELIMITER

This attribute is not used.

#### ROW\_DELIMITER

This attribute is not used.

#### REC\_DELIMITER

This attribute is not used.

#### FORMAT

Specifies the way in which the source data is formatted. Valid values are INTERNAL\_DIM\_XML, EXTERNAL\_DIM\_XML, BINARY (a proprietary format), STRATIFY, and PERL. Each value has the following meaning:

- INTERNAL\_DIM\_XML indicates the .xml file was created using Developer Studio and therefore conforms to dimensions.dtd.
- EXTERNAL\_DIM\_XML indicates the .xml file was read in from an externally managed taxonomy. The TRANSFORMER sub-element specifies the .xsl stylesheet that converts .xml from an externally managed taxonomy to Endeca-compatible .xml. (The pipeline developer authors an .xsl stylesheet that converts the .xml to conform to external\_dimensions.dtd.)

- STRATIFY indicates the .xml file was read in from a Stratify Taxonomy Manager. The TRANSFORMER sub-element converts the file from Stratify .xml to Endeca-compatible .xml. (Endeca provides an .xsl stylesheet that converts the .xml to conform to external\_dimensions.dtd.)
- To use Perl code in a DIMENSION\_ADAPTER, you must configure the PASS\_THROUGH element with additional attribute values and also provide the code itself as the PCDATA of PASS\_THROUGH.

### REQUIRE\_DATA

Set to TRUE by default, this attribute causes an error to generate if the URL does not exist or is empty. The attribute needs to be declared within a dimension adapter only if it is being set to FALSE (to turn the error off).

### COMPRESSION\_LEVEL

Controls the level of data compression when an adapter is configured for OUTPUT. The default compression level for this attribute is 0 (no compression). The value -1 for this attribute has been deprecated. Values 1-9 indicate increasing levels of compression.

### Sub-elements

The following table provides a brief overview of the DIMENSION\_ADAPTER sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <code>&lt;!-- ... --&gt;</code> .
DIMENSION_SOURCE	Specifies the name of a dimension adapter from which the component should read dimensions.
TRANSFORMER	Transforms source .xml data into Endeca-compatible .xml dimensions according to a specified .xslt style sheet.
PASS_THROUGH	In a DIMENSION_ADAPTER, the PASS_THROUGH element provides a means to run Perl code. Running Perl code in a dimension adapter is not supported.

### Example

This example shows two DIMENSION\_ADAPTER elements inputting dimension XML files to the pipeline.

```
<DIMENSION_ADAPTER URL="Dimension.xml" NAME="Dimensions" FORMAT="INTERNAL_DIM_XML"
DIRECTION="INPUT" />
<DIMENSION_ADAPTER URL="WineTypeDimension.xml" NAME="TypeDimension" FORMAT="INTERNAL_DIM_XML"
DIRECTION="INPUT" />
<DIMENSION_SERVER NAME="DimensionServer" FORMAT="INTERNAL_DIM_XML" PER-
SIST_URL="../partition0/state/AutogenDimensions.xml">
  <DIMENSION_SOURCE>Dimensions</DIMENSION_SOURCE>
  <DIMENSION_SOURCE>TypeDimension</DIMENSION_SOURCE>
</DIMENSION_SERVER>
```

## DIMENSION\_SERVER

A DIMENSION\_SERVER element provides a single point of reference for other components in Pipeline.epx to access dimensions.

The DIMENSION\_SERVER must specify all necessary input DIMENSION\_ADAPTER elements as its sources.

## DTD

```

<!ELEMENT DIMENSION_SERVER
  ( COMMENT?
  , DVAL_ID_LIMITS?
  , DIMENSION_SOURCE+
  , NAV_CONFIG_SOURCE?
  )
>
<!ATTLIST DIMENSION_SERVER
  NAME                CDATA                #REQUIRED
  PERSIST_URL         CDATA                #IMPLIED
  FORMAT              (XML | INTERNAL_DIM_XML | BINARY) #IMPLIED
  MATCH_COUNT_LOG    CDATA                #IMPLIED
  %compression;
>

```

## Attributes

The following sections describe the DIMENSION\_SERVER element's attributes.

### NAME

Identifies the DIMENSION\_SERVER so that it can be referenced by other pipeline components.

### PERSIST\_URL

Specifies the location of the persistent dimension data created by auto-generation. The path is either an absolute path, or a relative path. Relative paths are relative to the location of the Pipeline.epx file. With an absolute path the protocol may be specified in RFC 2396 syntax. Usually this means the prefix file:/// precedes the path to the data file.

### FORMAT

Specifies the file format of the PERSIST\_URL value. There are two valid values, INTERNAL\_DIM\_XML or BINARY. INTERNAL\_DIM\_XML indicates the file is stored as .xml that conforms to Endeca's .dtds. BINARY indicates the file is stored as a proprietary format that only the Endeca system can read and write.



**Note:** The XML format has been deprecated in favor of the INTERNAL\_DIM\_XML format.

### MATCH\_COUNT\_LOG

Provides a name reference to a LOG component that tracks the number of dimension value to property value matches and writes them to log file.



**Note:** This attribute is deprecated and will be removed in a future release. See "Endeca Logging and Reporting System Overview" in the *Endeca Administration Guide* for details about how to implement logging in your Endeca application.

### COMPRESSION\_LEVEL

Controls the level of data compression when an adapter is configured for OUTPUT. The default compression level for this attribute is 0 (no compression). The value -1 for this attribute has been deprecated. Values 1-9 indicate increasing levels of compression.

## Sub-elements

The following table provides a brief overview of the DIMENSION\_SERVER sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <code>&lt;!-- ... --&gt;</code> .
DVAL_ID_LIMITS	Specifies the minimum and maximum dimension value IDs that Forge can assign during auto generation.
DIMENSION_SOURCE	Specifies the name of a dimension adapter from which the component should read dimensions

### Example

This example shows a DIMENSION\_SERVER element that contains sources naming two DIMENSION\_ADAPTER inputs to the pipeline. Both sources are in .xml format.

```
<DIMENSION_SERVER NAME="DimensionServer" FORMAT="INTERNAL_DIM_XML" PERSIST_URL="
../partition0/state/AutogenDimensions.xml">
  <DVAL_ID_LIMITS MIN="10000000" MAX="2147483647"/>
  <DIMENSION_SOURCE>Dimensions</DIMENSION_SOURCE>
  <DIMENSION_SOURCE>TypeDimension</DIMENSION_SOURCE>
</DIMENSION_SERVER>
```

## DIMENSION\_SOURCE

A DIMENSION\_SOURCE element specifies the name of a dimension adapter from which a component should read dimensions.

For example, several DIMENSION\_SOURCE elements may be nested within DIMENSION\_SERVER and name several DIMENSION\_ADAPTER elements as input to the Pipeline.epx file.

### DTD

```
<!ELEMENT DIMENSION_SOURCE (#PCDATA)>
```

### Attributes

The DIMENSION\_SOURCE element has no attributes.

### Sub-elements

The DIMENSION\_SOURCE element has no sub-elements.

### Example

This example shows two DIMENSION\_SOURCE elements naming two DIMENSION\_ADAPTER elements as input.

```
<DIMENSION_ADAPTER URL="Dimension.xml" NAME="Dimensions" FORMAT="XML" DIRECTION="IN-
PUT"/>
<DIMENSION_ADAPTER URL="WineTypeDimension.xml" NAME="TypeDimension" FORMAT="XML"
DIRECTION="INPUT"/>
<DIMENSION_SERVER NAME="DimensionServer" FORMAT="XML" PERSIST_URL="../parti-
tion0/state/AutogenDimensions.xml">
  <DIMENSION_SOURCE>Dimensions</DIMENSION_SOURCE>
```

```
<DIMENSION_SOURCE>TypeDimension</DIMENSION_SOURCE>
</DIMENSION_SERVER>
```

## DVAL\_ID\_LIMITS

A DVAL\_ID\_LIMITS element specifies the minimum and maximum dimension value IDs that Forge can assign during auto generation.

### DTD

```
<!ELEMENT DVAL_ID_LIMITS
  ( COMMENT?
  )
>
<!ATTLIST DVAL_ID_LIMITS
  MIN      CDATA      #REQUIRED
  MAX      CDATA      #REQUIRED
>
```

### Attributes

The following sections describe the DVAL\_ID\_LIMITS element's attributes.

#### MIN

Indicates the minimum value that Forge can assign to a dimension value when auto generating IDs.

#### MAX

Indicates the maximum value that Forge can assign to a dimension value when auto generating IDs.

### Sub-elements

The following table provides a brief overview of the DVAL\_ID\_LIMITS sub-element.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.

### Example

This example shows a DVAL\_ID\_LIMITS element constraining dimension value IDs to between 10000000 and 2147483647.

```
<DIMENSION_SERVER NAME="DimensionServer" FORMAT="XML" PERSIST_URL=" ../parti-
tion0/state/AutogenDimensions.xml ">
  <DVAL_ID_LIMITS MIN="10000000" MAX="2147483647"/>
  <DIMENSION_SOURCE>Dimensions</DIMENSION_SOURCE>
  <DIMENSION_SOURCE>TypeDimension</DIMENSION_SOURCE>
</DIMENSION_SERVER>
```

## EXPRBODY

An EXPRBODY element contains Perl code that manipulates data. Perl code is often useful if a data manipulation task is too complicated to perform in XML. EXPRBODY is a child of EXPRESSION.

**DTD**

```
<!ELEMENT EXPRBODY (#PCDATA)>
```

**Attributes**

The EXPRBODY element has no attributes.

**Sub-elements**

The EXPRBODY element has no sub-elements.

**Example**

This example shows the outline of an expression using Perl.

```
<EXPRESSION TYPE="VOID" NAME="PERL">
  <EXPRBODY>
    ...PERL code here...
  </EXPRBODY>
</EXPRESSION>
```

**EXPRESSION**

An EXPRESSION element instructs Forge about how to modify records.

EXPRESSION elements appear within record manipulators to specify the property values, dimension values, and records that are to be modified, and how to modify them. An expression consists of an EXPRESSION element with TYPE and NAME attributes. Expressions may contain EXPRNODE elements, which have NAME and VALUE attributes, to supply additional configuration information. There are several dozen individual expressions distinguished by their TYPE and NAME attributes. Each of these expressions is described in the *Endeca Data Foundry Expression Reference*.

Expressions may also contain other expressions; the contained expressions may provide values used by the containing expression, or the containing expression may provide control over which of the contained expressions are evaluated. See the *Endeca Data Foundry Expression Reference* for the full list of available expressions.



**Note:** Oracle recommends that you perform record manipulation with the PERL\_MANIPULATOR element rather than with the EXPRESSION and RECORD\_MANIPULATOR elements. However, if you need to access and modify dimension sources (e.g., a dimension adapter or dimension server), you should still use expressions such as DVAL PERL. The Perl manipulator does not access dimension sources.

**DTD**

```
<!ELEMENT DIMENSION_ADAPTER
  ( COMMENT?
  , EXPRBODY?
  , (EXPRNODE | EXPRESSION)*
  )
>
<!ATTLIST EXPRESSION
  TYPE ( PROPERTY
        DVAL
        INTEGER
        STRING
        STREAM
        VOID
        FLOAT ) #REQUIRED
```

NAME	CDATA	#REQUIRED
LABEL	CDATA	#IMPLIED
URL	CDATA	#IMPLIED

## Attributes

The following sections describe the EXPRESSION attributes.

### TYPE

Describes the return value for the expression. For example, a FLOAT expression returns a floating point number. The valid values for TYPE are as follows: PROPERTY, DVAL, INTEGER, STRING, VOID, and FLOAT.



**Note:** STREAM is used internally by the MDEX Engine.

### NAME

Describes the operation being performed. Expressions are typically referred to by the combination of their TYPE and NAME values (for example DVAL CONST). This combination helps to distinguish cases where there are several expressions with different TYPE values but the same NAME value (for example, DVAL CONST, FLOAT CONST, and INTEGER CONST). The *Endeca Data Foundry Expression Reference* contains a help topic for each NAME expression available.

### URL

Used in Perl expressions. Specifies the URL (file) from which an expression can read Perl code. The code can be up to 65534 characters long.

## Sub-elements

The following table provides a brief overview of the EXPRESSION sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.
EXPRBODY	Contains Perl code that manipulates data.
EXPRNODE	Provides a generic way of sending a variety of information to an EXPRESSION.
EXPRESSION	Instructs Forge about how to modify data in the Pipeline.epx file.

## Example

This example shows a mathematical expression that adds two constant values (5 and 6).

```
<EXPRESSION TYPE="INTEGER" NAME="MATH">
  <EXPRNODE NAME="TYPE" VALUE="INTEGER" />
  <EXPRNODE NAME="OPERATOR" VALUE="ADD" />
  <EXPRESSION TYPE="INTEGER" NAME="CONST">
    <EXPRNODE NAME="VALUE" VALUE="5" />
  </EXPRESSION>
  <EXPRESSION TYPE="INTEGER" NAME="CONST">
    <EXPRNODE NAME="VALUE" VALUE="6" />
  </EXPRESSION>
</EXPRESSION>
```

This example adds a dimension value ID to the current record. The dimension value ID is determined by the mapping between the value of the P\_Score property in the source data and the dimension values contained in the dimension with ID equal to 9.

```
<EXPRESSION NAME="ADD_DVAL" TYPE="VOID">
  <EXPRESSION NAME="MATCH" TYPE="DVAL">
    <EXPRNODE NAME="DIMENSION_ID" VALUE="9" />
    <EXPRESSION NAME="IDENTITY" TYPE="PROPERTY">
      <EXPRNODE NAME="PROP_NAME" VALUE="P_Score" />
    </EXPRESSION>
  </EXPRESSION>
</EXPRESSION>
```

## EXPRNODE

An EXPRNODE element provides a generic way of sending a variety of information to an EXPRESSION.

The information could be descriptions, data types, constant values (parameters), and so on. Comparatively speaking, this information is similar to a parameter for a function. See the *Endeca Data Foundry Expression Reference* for the full list of available expressions.

### DTD

```
<!ELEMENT EXPRNODE (EXPRNODE*)>
<!ATTLIST EXPRNODE
  NAME          CDATA          #REQUIRED
  VALUE         CDATA          #IMPLIED
>
```

### Attributes

The following sections describe the EXPRNODE attributes.

#### NAME

Describes the EXPRNODE element. Because EXPRNODE can be so broadly used to modify an expression, the NAME attribute varies in relation to the expression that it modifies. For example, NAME can specify a variety of values such as TYPE, NAME, OPERATOR, AUTO\_GEN, OPERATION, and so on. See an expression's help topic for details about how the NAME attribute of an expression node modifies an expression.

#### VALUE

Provides a value that corresponds to the NAME attribute. Because EXPRNODE can be so broadly used, the VALUE attribute can specify a variety of values. For example, INTEGER may correspond to TYPE; SUM may correspond to NAME; ADD may correspond to OPERATOR, and so on. See an expression's help topic for details about how the VALUE attribute of an expression node modifies an expression.

### Sub-elements

The EXPRNODE element can contain additional EXPRNODE elements as sub-elements.

### Example

This example shows an expression adding a dimension value ID to the current record. The second and third expressions use EXPRNODE to provide name and value information for the parent EXPRESSION.

```
<EXPRESSION NAME="ADD_DVAL" TYPE="VOID">
  <EXPRESSION NAME="MATCH" TYPE="DVAL">
    <EXPRNODE NAME="DIMENSION_ID" VALUE="9" />
```

```

    <EXPRESSION NAME="IDENTITY" TYPE="PROPERTY">
      <EXPRNODE NAME="PROP_NAME" VALUE="P_Score" />
    </EXPRESSION>
  </EXPRESSION>
</EXPRESSION>

```

## ID\_SERVER

An ID\_SERVER element specifies the location and format of the id-server.xml file.

The path is either an absolute path or a path relative to the location of the Pipeline.epx file. With an absolute path, the protocol must be specified in RFC 2396 syntax. Usually this means the prefix `file:///` precedes the path to the data file. Relative URLs must not specify the protocol and are relative to the URL used to locate the Pipeline.epx file.



**Note:** The ID\_SERVER element has been deprecated.

### DTD

```

<!ELEMENT ID_SERVER EMPTY>
<!ATTLIST ID_SERVER
  URL          CDATA          #REQUIRED
  FORMAT       CDATA          #REQUIRED
  REQUIRE_DATA (TRUE | FALSE) "FALSE"
>

```

### Attributes

The following sections describe the ID\_SERVER element's attributes.

#### URL

Specifies the location of the id-server file. The path is either an absolute path or a path relative to the location of the Pipeline.epx file. With an absolute path, the protocol must be specified in RFC 2396 syntax. Usually this means the prefix `file:///` precedes the path to the data file. Relative URLs must not specify the protocol and are relative to the URL used to locate the Pipeline.epx file.

#### FORMAT

Specifies the file format of id-server file. This value may be either XML or BINARY.

#### REQUIRE\_DATA

Indicates whether the id-server.xml file must exist. If set to TRUE, Forge throws an error if the file is missing. If set to FALSE, Forge creates the file. The default is FALSE.

### Sub-elements

The ID\_SERVER element has no sub-elements.

### Example

This example shows the id-server.xml file but does not require that it contain data.

```

<PIPELINE NAME="Sample Wine Pipeline">
  <LOG URL="../../../logs/pipeline.log" NAME="LogFile" TYPE="UNIQUE" />
  <ID_SERVER URL="../../../partition0/state/id-server.xml"
    FORMAT="XML" REQUIRE_DATA="FALSE" />

```

```
...Remainder of pipeline removed for simplicity...
</PIPELINE>
```

## INDEXER\_ADAPTER

An INDEXER\_ADAPTER element writes records, dimensions, and index configuration information for further processing by Dgidx (the indexer).

For Dgidx to process output from the indexer adapter, the output format attributes must be configured as follows:

- Set the OUTPUT\_RECORD\_FORMAT to BINARY.
- Set the OUTPUT\_DIMENSION\_FORMAT attributes to XML.

Dgidx cannot process any other combinations of formats.

### DTD

```
<!ELEMENT INDEXER_ADAPTER
  ( COMMENT?
  , RECORD_SOURCE
  , DIMENSION_SOURCE*
  , NAV_CONFIG_SOURCE?
  , PASS_THROUGH*
  , EXPRESSION*
  , RECORD_ID_SPEC?
  , RECORD_GROUP?
  , ROLLOVER?
  )
>
<!ATTLIST INDEXER_ADAPTER
  NAME CDATA #REQUIRED
  FRC_PVAL_IDX (TRUE | FALSE) "TRUE"
  OUTPUT_URL CDATA #IMPLIED
  OUTPUT_PREFIX CDATA #IMPLIED
  OUTPUT_RECORD_FORMAT (XML|BINARY) "BINARY"
  OUTPUT_DIMENSION_FORMAT (XML|INTERNAL_DIM_XML) "INTERNAL_DIM_XML"
  MULTI (TRUE|FALSE) #IMPLIED
  MULTI_PROP_NAME CDATA #IMPLIED
  FILTER_UNKNOWN_PROPS (TRUE|FALSE) "TRUE"
  %compression;
```

### Attributes

The following sections describe the INDEXER\_ADAPTER element's attributes.

#### NAME

Identifies the INDEXER\_ADAPTER so that it can be referenced by other components in Pipeline.epx.

#### FRC\_PVAL\_IDX

This attribute forces Forge to create a record or property index for better performance during processing. TRUE creates an index. FALSE lets Forge create indices when necessary. The default is TRUE.

#### OUTPUT\_URL

Identifies the directory to which Forge writes its files and processed records. The path is either an absolute path or a path relative to the location of the Pipeline.epx file. With an absolute path, the protocol must be

specified in RFC 2396 syntax. Usually this means the prefix `file:///` precedes the path to the data file. Relative URLs must not specify the protocol and are relative to the URL used to locate the Pipeline.epx file.

### OUTPUT\_PREFIX

Specifies the filename prefix to add when Forge writes its output files. The default is `out`.

### OUTPUT\_RECORD\_FORMAT

Specifies the output format of the records Forge creates. The default is `BINARY`. Although `BINARY` is the most common output, it may be useful for troubleshooting to set the value to `XML` if you need to visually inspect the indexed records.

### OUTPUT\_DIMENSION\_FORMAT

Specifies the output format of the dimensions file that Forge creates. The default value is `INTERNAL_DIM_XML`.



**Note:** The `XML` value is no longer used.

### MULTI

Specifies whether multiple output files should be created. If `MULTI="TRUE"` then Forge checks `MULTI_PROP_NAME` for the location.

### MULTI\_PROP\_NAME

Specifies a property name that indicates the location that a record coming through Forge should be written to. (Forge stores a property on each record that is the source URL for that record.) Setting this property is useful if you want to process multiple input files and also keep the output distinct. In such cases, set `MULTI="TRUE"` and then set `MULTI_PROP_NAME="propname"` to specify the location that the given record coming through Forge should be written to.

For example, suppose you have an input files named `datedfile#1.txt` and `datedfile#2.txt` that are unique based on the date in the file name. Further suppose you set `MULTI="TRUE"`, set `MULTI_PROP_NAME="Absolute_url"`, and set `OUTPUT_PREFIX="out"` in an `OUTPUT` adapter. Forge processes the files from the `Absolute_url` value of the input directory and file name, and Forge writes `out.datedfile#1.records.xml` and `out.datedfile#2.records.xml` to the `OUTPUT_URL`. The output files contain only the processed records from their corresponding input files.

### FILTER\_UNKNOWN\_PROPS

Specifies whether unmapped properties should be removed from the processed records. The default value is `TRUE`. If you are using a partial update pipeline, this value should be the same in both the baseline pipeline and the partial update pipeline.

### COMPRESSION\_LEVEL

Controls the level of data compression when the indexer adapter writes record data. The default compression level for this attribute is 0 (no compression). The value -1 for this attribute has been deprecated. Values 1-9 indicate increasing levels of compression.

### Sub-elements

The following table provides a brief overview of the `INDEXER_ADAPTER` sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <code>&lt;!-- ... --&gt;</code> .

Sub-element	Brief description
RECORD_SOURCE	Specifies the name of a pipeline component from which this component should read records.
DIMENSION_SOURCE	Specifies the name of a pipeline component from which the component should read dimensions.
PASS_THROUGH	This sub-element is no longer used as part of an indexer adapter.
EXPRESSION	This sub-element is no longer used as part of an indexer adapter.
RECORD_GROUP	Provides property and dimension keys for record grouping.
ROLLOVER	Controls the point at which record output gets rolled over to a new file, and which Dgidx (indexer) processes records.

### Example

This example shows an indexer adapter configured with most of the default formats and elements that point to the source files for records, dimensions, and index configuration information.

```
<INDEXER_ADAPTER NAME="IndexerAdapter
  OUTPUT_URL=" ../partition0/forge_output/"
  FRC_PVAL_IDX="TRUE" OUTPUT_PREFIX="wine"
  OUTPUT_RECORD_FORMAT="BINARY"
  OUTPUT_DIMENSION_FORMAT="INTERNAL_DIM_XML">
  <RECORD_SOURCE>RecordManipulator</RECORD_SOURCE>
  <DIMENSION_SOURCE>DimensionServer</DIMENSION_SOURCE>
</INDEXER_ADAPTER>
```

## JAVA\_MANIPULATOR

A JAVA\_MANIPULATOR element executes code in a Java class to perform data manipulation on properties and records.

You can write your own custom Java class and have it executed with this component. Note that a pipeline can have multiple Java manipulators.

### DTD

```
<!ELEMENT JAVA_MANIPULATOR
  ( COMMENT?
  , RECORD_SOURCE
  , PASS_THROUGH*
  )
>
<!ATTLIST JAVA_MANIPULATOR
  NAME          CDATA          #REQUIRED
  CLASS_NAME    CDATA          #IMPLIED
  JAVA_HOME     CDATA          #IMPLIED
  JAVA_CLASSPATH CDATA          #IMPLIED
>
```

### Attributes

The following sections describe the JAVA\_MANIPULATOR element's attributes.

#### NAME

Identifies the JAVA\_MANIPULATOR so that it can be referenced by other components in the pipeline.

### CLASS\_NAME

Specifies a Java class name. When the Java manipulator runs, it loads and runs the specified class. The .jar file must be accessible via the file system on the machine where Forge is running.

If you place the class .jar file in a location other than the ENDECA\_ROOT/lib/java directory, use the JAVA\_CLASSPATH attribute to specify the location of the .jar file.

### JAVA\_HOME

Specifies the location of the Java runtime engine (JRE). If this attribute is not specified, the Java manipulator sets the location by looking for a setting in the following order:

1. The Forge --javaHome command-line parameter.
2. The ENDECA\_ROOT\j2sdk directory.
3. The JAVA\_HOME environment variable.

An error is logged if a Java home could not be found.

### JAVA\_CLASSPATH

Specifies the absolute path to the .jar file on every machine on which Forge runs. The .jar file must contain the class (and all other classes it depends on) specified in the CLASS\_NAME attribute. If you use the Forge --javaClasspath flag, it overrides the JAVA\_CLASSPATH attribute in the pipeline.

The default classpath points to all the .jar files in the ENDECA\_ROOT/lib/java directory. If either the JAVA\_CLASSPATH attribute or the Forge --javaClasspath flag is used, the specified classpath is prefixed to the default classpath.

### Sub-elements

The following table provides a brief overview of the JAVA\_MANIPULATOR sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.
RECORD_SOURCE	Specifies the name of a pipeline component from which this component should read records.
PASS_THROUGH	Provides additional parameters for the Java class.

### Example

This example shows a Java manipulator that will execute the TermExtractor class, with four parameters being passed in.

```
<JAVA_MANIPULATOR
  CLASS_NAME="com.endeca.edf.termextractor.TermExtractor"
  NAME="TermExtractor">
  <RECORD_SOURCE>LoadAllMainData</RECORD_SOURCE>
  <PASS_THROUGH NAME="RECORD_SPEC_PROP_NAME">P_WineID</PASS_THROUGH>
  <PASS_THROUGH NAME="TEXT_PROP_NAME">P_Description</PASS_THROUGH>
  <PASS_THROUGH NAME="OUTPUT_PROP_NAME">P_Terms</PASS_THROUGH>
  <PASS_THROUGH NAME="ALL_TERMS_OUTPUT_PROP_NAME">P_AllTerms</PASS_THROUGH>
</JAVA_MANIPULATOR>
```

## JOIN\_ENTRY

A JOIN\_ENTRY element specifies the key used to join records from a RECORD\_SOURCE.

For more information about joining source data, see the *Endeca Forge Guide*.

### DTD

```
<!ELEMENT JOIN_ENTRY
  ( RECORD_SOURCE
    ; KEY_DIMENSION*
  )
>
```

### Attributes

The JOIN\_ENTRY element has no attributes.

### Sub-elements

The following table provides a brief overview of the JOIN\_ENTRY sub-elements.

Sub-element	Brief description
RECORD_SOURCE	Specifies the name of a pipeline component from which this component should read records.
KEY_DIMENSION	Determines how records are to be compared by the join implementation.

### Example

This example shows the join entries of three record sources being joined according to SID and ID keys.

```
<RECORD_JOIN JOIN_TYPE="LEFT_JOIN">
  <JOIN_ENTRY>
    <RECORD_SOURCE>rs-1</RECORD_SOURCE>
    <KEY_DIMENSION TYPE="PVAL" ID="sid"/>
  </JOIN_ENTRY>
  <JOIN_ENTRY>
    <RECORD_SOURCE>rs-2</RECORD_SOURCE>
    <KEY_DIMENSION TYPE="PVAL" ID="id"/>
  </JOIN_ENTRY>
  <JOIN_ENTRY>
    <RECORD_SOURCE>rs-3</RECORD_SOURCE>
    <KEY_DIMENSION TYPE="PVAL" ID="id"/>
  </JOIN_ENTRY>
</RECORD_JOIN>
```

## PASS\_THROUGH

A PASS\_THROUGH element provides additional configuration information for a pipeline component.

A PASS\_THROUGH element has many distinct functions depending on the format of your source data. The source data can be delimited, fixed width, Perl code, an ODBC or JDBC datasource, or a custom record adapter.

- For delimited source files that do not have a header row in the file, PASS\_THROUGH specifies the header names. Provide a delimited string containing each header name for the source data. See the COL\_DELIMITER attribute of the RECORD\_ADAPTER for the appropriate delimiter of the string.

- For fixed-width source files, PASS\_THROUGH specifies how to handle reading source data. For information on using PASS\_THROUGH when reading fixed-width data, see the Endeca Developer's Studio Help.
- To run Perl code in expressions, the PASS\_THROUGH element specifies the Perl code to run as its PCDATA.
- To extract source data from an ODBC datasource, use PASS\_THROUGH elements specifies the SQL code to run to identify the DSN and the user name and password of the datasource.
- To extract source data from a JDBC datasource, use PASS\_THROUGH elements to specify the JDBC driver, the database URL, and SQL to execute against the database. If necessary, you can create additional PASS\_THROUGH sub-elements that specify user name and password parameters. Note that configuring user name and password parameters varies according to your JDBC driver. For example, with Oracle JDBC Thin drivers, the user name and password parameters are included as part of the DB\_URL string rather than as separate PASS\_THROUGH sub-elements. You may have to refer to the documentation for your JDBC driver type to determine exact configuration requirements. See the Example section below.
- For an EXCHANGE adapter, the PASS\_THROUGH element specifies the a path to the Microsoft Exchange server and if authentication is required, PASS\_THROUGH specifies a path to a key\_ring.xml file. See the Example section below.
- For a custom record adapter of FORMAT type JAVA\_ADAPTER, the adapter developer may choose to expose datasource parameters that a pipeline author can provide in PASS\_THROUGH elements. See the *Endeca Content Adapter Developer's Guide* for more information.
- For a JAVA\_MANIPULATOR, the pipeline author can provide parameters to the Java class.

## DTD

```
<!ELEMENT PASS_THROUGH (#PCDATA)>
<!ATTLIST PASS_THROUGH
  NAME      CDATA      #REQUIRED
>
```

## Attributes

The following section describes the PASS\_THROUGH element's attributes.

### NAME

Specifies a variety of parameters that correspond to the function of the PASS\_THROUGH element in its parent element. The following values are possible:

- For delimited source files, NAME specifies a descriptive name for the header row. For fixed-width source files, you must use several PASS\_THROUGH elements, each of which has a different NAME value. See "Preparing Your Source Data" in the Endeca Developer Studio Help.
- To run Perl code, NAME must be set to PERL.
- To run SQL code, NAME must be set to SQL.
- To pass parameters in a JAVA\_MANIPULATOR to a Java class, NAME must be set to the name of the parameter.
- To access a Microsoft Exchange Server, NAME must be set to URL. The value of the NAME attribute is a path to the server. If authentication is required, then a second element is necessary where NAME must be set to KEY\_RING. The value of the NAME attribute is a path to the key\_ring.xml file. See the Example below.
- To access an ODBC data source, one PASS\_THROUGH element must have NAME attribute set to SQL. Then a second PASS\_THROUGH element is required to identify the DSN, user name, and password. Its NAME attribute must be set to DSN. Provide the DSN, user name, and password values as a semi-colon (;) separated list. You may need to refer to your data source documentation for the correct key values of user name and password. In many ODBC data sources, these key values are UID and PWD. See the Example section below.

- To access a JDBC data source, one PASS\_THROUGH element must have NAME set to DB\_DRIVER\_CLASS where DB\_DRIVER\_CLASS is the fully qualified Java class name of the JDBC database driver to use. In another PASS\_THROUGH element, NAME must be set to DB\_URL (where DB\_URL is the URL of the database to connect to).

### Sub-elements

The PASS\_THROUGH element has no sub-elements.

### Examples

This example shows how a record adapter's PASS\_THROUGH sub-element might be configured if the wine\_data.txt file had no header information.

```
<RECORD_ADAPTER
  URL=" ../incoming/wine_data.txt " NAME="LoadMainData "
  FORMAT="DELIMITED"
  DIRECTION=" INPUT "
  COL_DELIMITER=" | "
  ROW_DELIMITER=" |\n" FILTER_EMPTY_PROPS="TRUE "
  <PASS_THROUGH
    NAME="HEADER_ROW">WINE_ID | YEAR | WINE | WINERY | PRICE | SCORE | REGION</PASS_THROUGH>
</RECORD_ADAPTER>
```

This example shows how a record adapter's PASS\_THROUGH sub-elements might be configured to access a Microsoft Exchange server.

```
<RECORD_ADAPTER
  NAME="Exchange_Server "
  DIRECTION=" INPUT "
  FORMAT=" EXCHANGE "
  JAVA_HOME=" C:\java ">
  <PASS_THROUGH
    NAME="URL">http://exchange.my.company.com/public</PASS_THROUGH>
  <PASS_THROUGH NAME="KEY_RING">path/key_ring.xml</PASS_THROUGH>
</RECORD_ADAPTER>
```

This example shows how a record adapter's PASS\_THROUGH sub-elements might be configured to access a JDBC datasource via an Oracle JDBC Thin driver.

```
<RECORD_ADAPTER
  NAME="JDBC_Datasource "
  DIRECTION=" INPUT "
  FORMAT=" JDBC "
  JAVA_HOME=" /usr/local/j2sdk "
  JAVA_CLASSPATH=" lib/oracle8i.zip "
  <PASS_THROUGH
    NAME="DB_DRIVER_CLASS">oracle.jdbc.driver.OracleDriver</PASS_THROUGH>
  <PASS_THROUGH NAME="DB_URL">
    jdbc:oracle:thin:username/password@127.0.0.1:1521:myOracleDatabase
  </PASS_THROUGH>
  <PASS_THROUGH
    NAME="SQL">SELECT * FROM DOCUMENTSET ORDER BY ID</PASS_THROUGH>
</RECORD_ADAPTER>
```

This example shows an input record adapter reading an ODBC datasource.

```
<RECORD_ADAPTER
  NAME="ODBC_Source "
  FORMAT=" ODBC "
  DIRECTION=" INPUT "
```

```

JAVA_HOME="C:\java">
<PASS_THROUGH
  NAME="DSN">Northwind;UID=john;PWD=sw0rdf1sh</PASS_THROUGH>
<PASS_THROUGH NAME="SQL">
  SELECT CustomerID, CompanyName, ContactName, ContactTitle, Address,
    City, Region, PostalCode, Country, Phone FROM Customers
</PASS_THROUGH>
</RECORD_ADAPTER>

```

## PERL\_MANIPULATOR

A PERL\_MANIPULATOR element executes Perl code to perform data manipulation on properties and records

You can provide Perl code to the PERL\_MANIPULATOR in any of the following three ways:

- Specify the code in the body of a PERL\_METHOD element. This approach is useful for simpler data manipulation and cases where you want to keep the Perl code in the Pipeline.epx file.
- Specify the code in a Perl file (.pl) external to your Pipeline.epx file, and identify the file in the URL attribute of a PERL\_METHOD element. This approach is useful if you want to maintain the Perl code outside the Pipeline.epx file or reuse the code by calling the file from more than one Pipeline.epx file.
- Specify the code in a Perl module (.pm), and identify the file in the CLASS\_NAME attribute of a PERL\_MANIPULATOR element. This approach is more useful in cases where the amount of Perl code is large or complex.

See the *Forge API Guide for Perl* for information about Perl methods available for data manipulation within the PERL\_MANIPULATOR and PERL\_METHOD elements.

### DTD

```

<!ELEMENT PERL_MANIPULATOR
  ( COMMENT?
  , RECORD_SOURCE*
  , RECORD_INDEX?
  , PERL_METHOD*
  )
>
<!ATTLIST PERL_MANIPULATOR
  NAME          CDATA      #REQUIRED
  CLASS_NAME    CDATA      #IMPLIED
>

```

### Attributes

The following sections describe the PERL\_MANIPULATOR element's attributes.

#### NAME

Identifies the PERL\_MANIPULATOR so that it can be referenced by other components in the pipeline. Do not use any spaces in the NAME of a Perl manipulator or the Perl code that Forge generates will be invalid.

#### CLASS\_NAME

Specifies a class name of a Perl module that you can create to call your Prepare, Next\_Record, Get\_Records, and Finish methods. When the Perl Manipulator runs, it loads and runs the specified class.

The Perl class must be located on the machine running Forge. It is convenient to locate the .pm file in the in the same location as other Perl modules for Endeca (ENDECA\_ROOT\lib\perl). Placing your .pm file in ENDECA\_ROOT\lib\perl does not require any additional configuration for Forge to locate it. However, if you upgrade Forge, you will have to copy this file to another location and copy it back in after upgrading.

If you place the file in a location other than `ENDECA_ROOT\lib\perl`, you must modify Perl's library search path to include the path to the `.pm` file. You can modify the path by either modifying your `PERLLIB` environment variable or by running Forge with the `--perllib` command-line option and providing the path as an argument

### Sub-elements

The following table provides a brief overview of the `PERL_MANIPULATOR` sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <code>&lt;!-- ... --&gt;</code> .
RECORD_SOURCE	Specifies the name of a pipeline component from which this component should read records.
RECORD_INDEX	Specifies a key used to identify records for record retrieval. This element is required if the Perl manipulator is using the <code>GET_RECORDS</code> attribute of <code>PERL_METHOD</code> .
PERL_METHOD	Specifies Perl code that Forge executes to perform data manipulation on properties and records.

### Examples

This example adds a record number property to each record as it is processed.

```
<PERL_MANIPULATOR NAME="PerlManipulator">
  <RECORD_SOURCE>LoadData</RECORD_SOURCE>
  <PERL_METHOD NAME="PREPARE">
    # Make sure we have exactly one record source
    if (scalar(@{ $this->record_sources }) != 1) {
      die("Perl Manipulator ", $this->name,
        " must have exactly one record source.");
    }
    # Keep the current record number in our context
    $this->context->{RECNO} = 0;
  </PERL_METHOD>
  <PERL_METHOD NAME="NEXT_RECORD">
    # Count this record
    ++$this->context->{RECNO};
    my $rec = $this->record_source(0)->next_record;
    # Careful: $rec will be undef if there are no more records
    if ($rec) {
      my $pval = new EDF::PVal("Record Number", $this->context->{RECNO});
      $rec->add_pvals($pval);
    }
    return $rec;
  </PERL_METHOD>
</PERL_MANIPULATOR>
```

This is another example that performs the same task of adding a record number property to each record; however, this example uses a subclass and the `CLASS_NAME` attribute to provide the code.

```
<PERL_MANIPULATOR NAME="PerlManip" CLASS_NAME="MyPerlManip">
  <RECORD_SOURCE>LoadData</RECORD_SOURCE>
</PERL_MANIPULATOR>
```

Where the contents of MyPerlManip.pm are as follows:

```
package MyPerlManip;
use EDF::Manipulator;
@ISA = qw(EDF::Manipulator);
sub new
{
    my $proto = shift;
    my $class = ref($proto) || $proto;
    my $this = {};
    bless($this, $class);
    $this->SUPER::init(@_);
    return $this;
}
sub prepare
{
    my $this = shift;
    if (scalar(@{ $this->record_sources }) != 1) {
        die("Perl Manipulator ", $this->name,
            " must have exactly one record source.");
    }
    $this->{RECNO} = 0;
}
sub next_record
{
    my $this = shift;
    ++$this->{RECNO};
    my $rec = $this->record_source(0)->next_record;
    if ($rec) {
        my $pval = new EDF::PVal("Record Number", $this->{RECNO});
        $rec->add_pvals($pval);
    }
    return $rec;
}
```

## PERL\_METHOD

A PERL\_METHOD element specifies Perl code that Forge executes to perform data manipulation on properties and records.

A PERL\_MANIPULATOR may contain up to four PERL\_METHOD elements. Each PERL\_METHOD element is distinguished by its NAME attribute which specifies when Perl code executes during record processing and how the PERL\_METHOD retrieves records. You can provide Perl code either in the body of the element, or you can point to a .pl file with the URL attribute.

### DTD

```
<!ELEMENT PERL_METHOD (#PCDATA)>
<!ATTLIST PERL_METHOD
    NAME          (PREPARE|FINISH|NEXT_RECORD|GET_RECORDS) #REQUIRED
    URL           CDATA #IMPLIED
    COMPRESSED    (TRUE|FALSE) #IMPLIED
    ENCODING      CDATA #IMPLIED
>
```

### Attributes

The following sections describe the PERL\_METHOD element's attributes.

## NAME

Specifies one of four possible values that describe either when the Perl code runs or the way in which the Perl code retrieves records. The Perl code associated with any of these attributes can be specified in the body of the element or in a .pl file. The valid values of this attribute are as follows:

- **PREPARE** — Specifies that the code within this element should execute at the beginning of record processing. PREPARE is useful for pre-processing tasks such as ensuring the correct number of data sources or ensuring the data sources have the correct properties and property values in the expected locations and so on.
- **NEXT\_RECORD** — Specifies that records should be returned from the RECORD\_SOURCE in sequential order. Returns the undef value when no additional records are available.
- **GET\_RECORDS** — Similar to NEXT\_RECORD but instead specifies that records should be returned from the RECORD\_SOURCE as identified by a record key. The key is specified in a RECORD\_INDEX sub-element of PERL\_MANIPULATOR.
- **FINISH** — Specifies that the code within this element should execute at the end of record processing. FINISH is useful for post-processing tasks, for example printing the total number of records processed.

## URL

Specifies a path to the Perl file (.pl) from which the PERL\_MANIPULATOR reads Perl code. This attribute is useful in cases where you prefer not to include the Perl code in the body of the PERL\_METHOD element, but instead want to write the code in a file that is external to the Pipeline.epx. If you point to a .pl file, you can also specify optional COMPRESSION and ENCODING attributes for the file.

The path is either an absolute path or a path relative to the location of the Pipeline.epx file. With an absolute path, the protocol must be specified in RFC 2396 syntax. Usually this means the prefix `file:///` precedes the path to the data file. Relative URLs must not specify the protocol and are relative to the URL used to locate the Pipeline.epx file.

## COMPRESSED

Specifies whether the Perl file being accessed is compressed. TRUE indicates the file is compressed and instructs Forge to uncompress it before processing. FALSE indicates the Perl file is not compressed. The default value is FALSE.

## ENCODING

Specifies the character encoding of the Perl file, indicated in the URL attribute. Forge supports the same encodings for the PERL\_MANIPULATOR and RECORD\_ADAPTER. See the ENCODING attribute of the RECORD\_ADAPTER for a full description of valid encodings.

## Sub-elements

The PERL\_METHOD element has no sub-elements.

## Example

This example includes three PERL\_METHOD elements, each of which refers to an individual .pl file to run the Perl code.

```
<PERL_MANIPULATOR NAME="PerlManipulator">
  <RECORD_SOURCE>LoadData</RECORD_SOURCE>
  <PERL_METHOD NAME="PREPARE" URL="prepare.pl"
    COMPRESSED="FALSE" ENCODING="ASCII">
  </PERL_METHOD>
  <PERL_METHOD NAME="NEXT_RECORD" URL="next_record.pl"
    COMPRESSED="FALSE" ENCODING="ASCII">
  </PERL_METHOD>
  <PERL_METHOD NAME="FINISH" URL="done.pl">
```

```

    COMPRESSED="FALSE" ENCODING="ASCII">
  </PERL_METHOD>
</PERL_MANIPULATOR>

```

## PIPELINE

A PIPELINE element is the root element of a project's Pipeline.epx file.

The PIPELINE element contains all the components, expressed as sub-elements, that describe and control the data transformation process. Pipeline.epx may contain components in any order. The entire PIPELINE is a repeatable sequence. Although components may appear in any order in the XML, they typically are ordered in a sequential flow to produce a "pipeline" feel. Actual execution flow is controlled by Forge and the way in which you connect components with naming references.

### DTD

```

<!ELEMENT PIPELINE
  (
    COMMENT?
    LOG*
    ID_SERVER
    RECORD_ADAPTER+
    RECORD_CACHE*
    RECORD_ASSEMBLER*
    RECORD_MANIPULATOR+
    PROP_MAPPER+
    PERL_MANIPULATOR+
    JAVA_MANIPULATOR+
    DIMENSION_ADAPTER+
    DIMENSION_SERVER
    INDEXER_ADAPTER
    UPDATE_ADAPTER
    SPIDER*
  )+
>
<!ATTLIST PIPELINE
  NAME      CDATA      #REQUIRED
>

```

### Attributes

The following section describes the PIPELINE element's attribute.

#### NAME

A unique name describing the pipeline.

### Sub-elements

The following table provides a brief overview of the PIPELINE sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.
LOG	Holds the information gathered by the INFO element. This element is deprecated.

Sub-element	Brief description
ID_SERVER	Specifies the location and format of the id-server.xml file. This element is deprecated.
RECORD_ADAPTER	Reads and writes record data in a variety of formats.
RECORD_CACHE	Sorts records according to a specified key and loads the records in memory for join operations.
RECORD_ASSEMBLER	Assembles (joins) records from multiple sources together based on a specified index.
RECORD_MANIPULATOR	Contains EXPRESSIONS, which are evaluated against each record as it flows through the pipeline. When an EXPRESSION is evaluated, it may modify the current record.
PROP_MAPPER	Maps properties from source data to Endeca properties and dimensions.
PERL_MANIPULATOR	Executes Perl code to perform data manipulation on properties and records.
JAVA_MANIPULATOR	Executes code in a Java class to perform data manipulation on properties and records.
DIMENSION_ADAPTER	Reads and writes dimension files in a variety of formats.
DIMENSION_SERVER	Provides both a single point of reference for other pipeline components to access dimensions.
INDEXER_ADAPTER	Writes records, dimensions, and navigation configuration information for further processing by Dgidx (the indexer).
UPDATE_ADAPTER	As part of a partial update pipeline, an UPDATE_ADAPTER element writes partial update information for a running Endeca MDEX Engine to perform a live update.
SPIDER	Adds the capability to crawl document hierarchies on a file system or on Web sites. This element is deprecated.

## Example

This example shows the sub-elements in a pipeline.

```
<PIPELINE NAME="Sample Wine Pipeline">
  <INDEXER_ADAPTER COMPRESSION_LEVEL="0" FILTER_UNKNOWN_PROPS="TRUE"
    FRC_PVAL_IDX="TRUE" MULTI_PROP_NAME="" NAME="IndexerAdapter"
    OUTPUT_DIMENSION_FORMAT="XML" OUTPUT_PREFIX="wine"
    OUTPUT_RECORD_FORMAT="BINARY" OUTPUT_URL="../partition0/forge_output/">
    <RECORD_SOURCE>PropDimMapper</RECORD_SOURCE>
    <DIMENSION_SOURCE>DimensionServer</DIMENSION_SOURCE>
    <ROLLOVER CUTOFF="2000000000" NAME="RECORD" NUM_IDX="1"
      PROP_NAME="" PROP_TYPE="ALPHA" REMOVE_PROP="FALSE" ROLL_URL=""
      TYPE="SIZE" VALIDATE="FALSE"/>
  </INDEXER_ADAPTER>
  <RECORD_ADAPTER COL_DELIMITER="|" DIRECTION="INPUT"
    FILTER_EMPTY_PROPS="TRUE" FORMAT="DELIMITED" FRC_PVAL_IDX="TRUE"
    NAME="LoadMainData" PREFIX="" REC_DELIMITER="" ROW_DELIMITER="|\n"
    URL="../incoming/wine_data.txt.gz">
    <COMMENT></COMMENT>
  </RECORD_ADAPTER>
  <DIMENSION_ADAPTER COL_DELIMITER="" DIRECTION="INPUT"
    FORMAT="INTERNAL_DIM_XML" NAME="Dimensions" REC_DELIMITER=""
    ROW_DELIMITER="" URL="dimensions.xml">
```

```

    <COMMENT></COMMENT>
  </DIMENSION_ADAPTER>
  <DIMENSION_SERVER NAME="DimensionServer"
    COMPRESSION_LEVEL="0" FORMAT="INTERNAL_DIM_XML"
    PERSIST_URL=" ../partition0/state/autogen_dimensions.xml ">
    <COMMENT></COMMENT>
    <DIMENSION_SOURCE>Dimensions</DIMENSION_SOURCE>
  </DIMENSION_SERVER>
  <PROP_MAPPER DEFAULT_MAP_MODE="NONE" DEFAULT_MAX_LENGTH=" "
    IMPLICIT_MAPPING="TRUE" NAME="PropDimMapper">
    <RECORD_SOURCE>LoadMainData</RECORD_SOURCE>
    <DIMENSION_SOURCE>DimensionServer</DIMENSION_SOURCE>
    <PROP_MAPPING_PROP MAX_LENGTH="0" PROP_NAME="P_Wine"
      TARGET_NAME="P_Name" />
    <PROP_MAPPING_PROP MAX_LENGTH="0" PROP_NAME="P_Description"
      TARGET_NAME="P_Description" />
    ...other PROP_MAPPER subelements removed for simplicity...
    <PROP_MAPPING_DIM MATCH_MODE="AUTO_GEN" MAX_LENGTH="0"
      PROP_NAME="P_Flavors" TARGET_NAME="Flavors" />
  </PROP_MAPPER>
</PIPELINE>

```

## PROP\_MAPPER

A PROP\_MAPPER element maps properties from source data to Endeca properties and dimensions.

The mappings dictate which dimension values are tagged to the current record and which property information is available for record search and display. Source data properties may be mapped implicitly if the source property names are identical to Endeca dimension names. However, implicit mapping cannot take place between source data properties and Endeca properties.

There should be exactly one PROP\_MAPPER element in a pipeline file. Use a single PROP\_MAPPER element to map source properties to Endeca properties and Endeca dimensions. (Creating two PROP\_MAPPER elements, one to map properties and one to map dimensions, causes errors for Dgidx.) Source properties may be mapped explicitly to Endeca properties and dimensions using PROP\_MAPPING\_PROP and PROP\_MAPPING\_DIM elements.

For details on the property mapper component, see the "Mapping Source Properties" chapter in the *Endeca Platform Services Forge Guide*.

### DTD

```

<!ELEMENT PROP_MAPPER
  (
    COMMENT?
    , RECORD_SOURCE
    , DIMENSION_SOURCE
    , RECORD_INDEX?
    , (PROP_MAPPING_NONE | PROP_MAPPING_PROP | PROP_MAPPING_DIM) *
  )
>
<!ATTLIST PROP_MAPPER
  NAME                CDATA                #REQUIRED
  IMPLICIT_MAPPING    (TRUE | FALSE)        "TRUE"
  DEFAULT_MAP_MODE    (NONE | PROP | DIM)    "NONE"
  DEFAULT_MAX_LENGTH  CDATA                #IMPLIED
>

```

## Attributes

The following sections describe the PROP\_MAPPER element's attributes.

### NAME

Identifies the PROP\_MAPPER so that it can be referenced by other components in the pipeline.

### IMPLICIT\_MAPPING

Specifies whether to create mappings between source properties and identically named Endeca dimensions. Implicit mapping cannot take place between source data properties and Endeca properties. Case sensitivity of the names matters. The default value is TRUE.

### DEFAULT\_MAP\_MODE

Specifies how to map properties that are neither explicitly mapped with the PROP\_MAPPING\_PROP or PROP\_MAPPING\_DIM elements nor implicitly mapped with the IMPLICIT\_MAPPING attribute. There are three valid values for this attribute: NONE, PROP, and DIM.

- NONE does not map properties that are explicitly or implicitly mapped. Unmapped properties are ignored during processing.
- PROP performs source property to Endeca property mapping. Forge does this by creating a PROP\_MAPPING\_PROP element, using the source property's name and value as the target Endeca property's name and value.
- DIM performs source property to Endeca dimension value mapping. Forge does this by creating a PROP\_MAPPING\_DIM element using the source property's name and value as the target dimension's name and dimension value.

### DEFAULT\_MAX\_LENGTH

Specifies the maximum number of characters that the value of a source property may contain before it is not mapped to an Endeca property or dimension value. Forge ignores (does not map) source properties larger than this value.

This attribute is useful to prevent mapping large source properties. For example, if you set this value to 20 and Forge encounters a source property value that is longer than 20 characters, Forge does not map the source property. You can override DEFAULT\_MAX\_LENGTH for any explicit mapping by specifying a new value in the MAX\_LENGTH attribute of PROP\_MAPPING\_DIM or PROP\_MAPPING\_PROP.

Setting DEFAULT\_MAX\_LENGTH or MAX\_LENGTH to either zero (0) or an empty string ("") disables a maximum setting and defaults to the maximum allowable values for property or dimension mapping. The default maximum value for property mapping is an unlimited length. The default maximum value for dimension value mapping is 255 characters.

## Sub-elements

The following table provides a brief overview of the PROP\_MAPPER sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.
RECORD_SOURCE	Specifies the name of a pipeline component from which this component should read records.
DIMENSION_SOURCE	Specifies the name of a pipeline component from which the component should read dimensions.

Sub-element	Brief description
RECORD_INDEX	Specifies a common key used to identify records during join operations, sort validation, and logging. When nested within PROP_MAPPER, the RECORD_INDEX element identifies records in log messages.
PROP_MAPPING_NONE	Prevents a source property from being mapped to an Endeca property or dimension.
PROP_MAPPING_PROP	Maps a source property to an Endeca property.
PROP_MAPPING_DIM	Maps a source property to an Endeca dimension.

## Examples

In this simple example, the PROP\_MAPPER uses its default of implicit mapping to map all source properties to dimensions with the same name.

```
<PROP_MAPPER NAME="mapper" DEFAULT_MAP_MODE="DIM">
  <RECORD_SOURCE>data</RECORD_SOURCE>
  <DIMENSION_SOURCE>DimensionServer</DIMENSION_SOURCE>
</PROP_MAPPER>
```

In this more complicated example, the PROP\_MAPPER maps the explicitly named source properties to named target properties and dimensions. All other source properties are mapped to dimensions of the same name because of the DEFAULT\_MAP\_MODE="DIM" setting. Source properties longer than 8 characters are not mapped..

```
<PROP_MAPPER NAME="mapper"
  DEFAULT_MAP_MODE="DIM"
  DEFAULT_MAX_LENGTH="8">
  <RECORD_SOURCE>data</RECORD_SOURCE>
  <DIMENSION_SOURCE>DimensionServer</DIMENSION_SOURCE>
  <PROP_MAPPING_PROP PROP_NAME="price" TARGET_NAME="Price"/>
  <PROP_MAPPING_DIM PROP_NAME="price" TARGET_NAME="Price Range"
    MATCH_MODE="NORMAL"/>
  <PROP_MAPPING_PROP PROP_NAME="sale_price" TARGET_NAME="Price"/>
  <PROP_MAPPING_DIM PROP_NAME="sale_price" TARGET_NAME="Price Range"
    MATCH_MODE="NORMAL"/>
</PROP_MAPPER>
```

## PROP\_MAPPING\_DIM

A PROP\_MAPPING\_DIM element maps a source property to an Endeca dimension.

In this element, you specify matching configuration attributes including the name of the source property, the name of the target dimension, and the matching mode. To map source properties to Endeca properties, use the PROP\_MAPPING\_PROP element.

For details on the property mapper component, see the "Mapping Source Properties" chapter in the *Endeca Platform Services Forge Guide*.

## DTD

```
<!ELEMENT PROP_MAPPING_DIM EMPTY>
<!ATTLIST PROP_MAPPING_DIM
  PROP_NAME CDATA #REQUIRED
  TARGET_NAME CDATA #REQUIRED
  MAX_LENGTH CDATA #IMPLIED
```

```
MATCH_MODE (NORMAL | MUST_MATCH | AUTO_GEN) "NORMAL"
>
```

### Attributes

The following sections describe the PROP\_MAPPING\_DIM element's attributes.

#### PROP\_NAME

Identifies the name of the source property to map.

#### TARGET\_NAME

Identifies the name of the Endeca dimension that the source property is mapped to. Forge tags the record with dimension values from the target dimension that match the source property value. If the TARGET\_NAME dimension does not exist, the property mapper creates it. The PROP\_MAPPER ensures that a DIMENSION\_REF and PRECEDENCE\_RULE exist for the TARGET\_NAME.

#### MAX\_LENGTH

Specifies the maximum number of characters a source property's value may contain before it is not mapped to an Endeca dimension value. In addition to not being mapped, properties with values longer than MAX\_LENGTH also generate a warning.

This attribute is useful to prevent mapping of large source properties. For example, if you set this value to 20 and Forge encounters a source property longer than 20 characters, Forge does not map that source property. Setting MAX\_LENGTH to either zero (0) or an empty string ("") disables a maximum setting and defaults to the maximum allowable value for dimension mapping. The default maximum value for dimension value mapping is 255 characters.

#### MATCH\_MODE

Specifies the match method for source data property. There are three valid values for this attribute: NORMAL, MUST\_MATCH, and AUTO\_GEN. The default value is NORMAL.

- NORMAL maps source property values that have a matching dimension value explicitly defined in the Dimension.xml file. Forge assigns the IDs specified in the Dimension.xml file to the Endeca records. Any source property values that do not have matching dimension values in the Dimension.xml file are ignored.
- MUST\_MATCH behaves exactly as NORMAL; however, if a property value does not have a match, a warning is issued.
- AUTO\_GEN automatically generates a dimension value name and ID for any source property value that does not have a matching dimension value in the Dimension.xml file.

### Sub-elements

The PROP\_MAPPING\_DIM element has no sub-elements.

### Example

This example shows several source properties being mapped to dimensions of a new name. Source properties longer than 8 characters are not mapped.

```
<PROP_MAPPER NAME="mapper" DEFAULT_MAP_MODE="DIM">
  <RECORD_SOURCE>data</RECORD_SOURCE>
  <DIMENSION_SOURCE>DimensionServer</DIMENSION_SOURCE>
  <PROP_MAPPING_PROP PROP_NAME="price" TARGET_NAME="Price"/>
  <PROP_MAPPING_DIM PROP_NAME="price" TARGET_NAME="Price Range"
    MAX_LENGTH="8" MATCH_MODE="NORMAL"/>
  <PROP_MAPPING_PROP PROP_NAME="sale_price" TARGET_NAME="Price"/>
  <PROP_MAPPING_DIM PROP_NAME="sale_price" TARGET_NAME="Price Range">
```

```

    MAX_LENGTH="8" MATCH_MODE="NORMAL" />
</PROP_MAPPER>

```

## PROP\_MAPPING\_NONE

A PROP\_MAPPING\_NONE element prevents a specified source property from being mapped to an Endeca property or dimension.

The PROP\_MAPPING\_NONE element provides a way to override mapping specified by the IMPLICIT\_MAPPING or DEFAULT\_MAP\_MODE attributes of the PROP\_MAPPER element. For example, you can set the PROP\_MAPPER element's DEFAULT\_MAP\_MODE attribute to DIM and then prevent dimension mapping for specified source properties using a PROP\_MAPPING\_NONE element.

For details on the property mapper component, see the "Mapping Source Properties" chapter in the *Endeca Platform Services Forge Guide*.

### DTD

```

<!ELEMENT PROP_MAPPING_NONE EMPTY>
<!ATTLIST PROP_MAPPING_NONE
    PROP_NAME    CDATA    #REQUIRED
>

```

### Attributes

The following section describes the PROP\_MAPPING\_NONE element's attribute.

#### PROP\_NAME

Identifies the source property that should not be mapped.

### Sub-elements

The PROP\_MAPPING\_NONE element has no sub-elements.

### Example

In this example, the reserve\_price source property is excluded from mapping.

```

<PROP_MAPPER NAME="mapper" DEFAULT_MAP_MODE="DIM">
  <RECORD_SOURCE>data</RECORD_SOURCE>
  <DIMENSION_SOURCE>DimensionServer</DIMENSION_SOURCE>
  <PROP_MAPPING_PROP PROP_NAME="price" TARGET_NAME="Price"/>
  <PROP_MAPPING_DIM PROP_NAME="price" TARGET_NAME="Price Range"
    MAX_LENGTH="8" MATCH_MODE="NORMAL"/>
  <PROP_MAPPING_PROP PROP_NAME="sale_price" TARGET_NAME="Price"/>
  <PROP_MAPPING_DIM PROP_NAME="sale_price" TARGET_NAME="Price Range"
    MAX_LENGTH="8" MATCH_MODE="NORMAL"/>
  <PROP_MAPPING_NONE PROP_NAME="reserve_price"/>
</PROP_MAPPER>

```

## PROP\_MAPPING\_PROP

A PROP\_MAPPING\_PROP element maps a source property to an Endeca property.

In the PROP\_MAPPING\_PROP element, you specify matching configuration attributes including the name of the source property, the name of the target property, and the maximum length of the source property's value to be mapped. To map source properties to Endeca dimensions, use the PROP\_MAPPING\_DIM element.

For details on the property mapper component, see the "Mapping Source Properties" chapter in the *Endeca Forge Guide*.

## DTD

```
<!ELEMENT PROP_MAPPING_PROP EMPTY>
<!ATTLIST PROP_MAPPING_PROP
  PROP_NAME    CDATA    #REQUIRED
  TARGET_NAME  CDATA    #REQUIRED
  MAX_LENGTH   CDATA    #IMPLIED
>
```

## Attributes

The following sections describe the PROP\_MAPPING\_PROP element's attributes.

### PROP\_NAME

Identifies the name of the source property to map.

### TARGET\_NAME

Identifies the name of the Endeca property that the source property value is mapped to. If the source and target properties have different names, Forge copies the source property value into the target property value. The PROP\_MAPPER ensures that a PROP\_REF exists for the TARGET\_NAME.

### MAX\_LENGTH

Specifies the maximum number of characters a source property's value may contain before it is not mapped to an Endeca property. In addition to not being mapped, properties with values longer than MAX\_LENGTH also generate a warning.

This attribute is useful to prevent mapping large source properties. For example, if you set this value to 20 and Forge encounters a source property with a value longer than 20 characters, Forge does not map the source property. Setting MAX\_LENGTH to either zero (0) or an empty string ("") disables a maximum setting and defaults to the maximum allowable value for property mapping. The default maximum value for property mapping is unlimited.

## Sub-elements

The PROP\_MAPPING\_PROP element has no sub-elements.

## Example

This example shows several source properties being mapped to Endeca properties of a new name. Source properties with values longer than 8 characters are not mapped.

```
<PROP_MAPPER NAME="mapper" DEFAULT_MAP_MODE="DIM">
  <RECORD_SOURCE>data</RECORD_SOURCE>
  <DIMENSION_SOURCE>DimensionServer</DIMENSION_SOURCE>
  <PROP_MAPPING_PROP PROP_NAME="price" TARGET_NAME="Price" MAX_LENGTH="8"/>
  <PROP_MAPPING_DIM PROP_NAME="price" TARGET_NAME="Price Range"
    MATCH_MODE="NORMAL" />
  <PROP_MAPPING_PROP PROP_NAME="sale_price" TARGET_NAME="Price"
    MAX_LENGTH="8" />
  <PROP_MAPPING_DIM PROP_NAME="sale_price" TARGET_NAME="Price Range"
    MATCH_MODE="NORMAL" />
</PROP_MAPPER>
```

## RECORD\_ADAPTER

A RECORD\_ADAPTER element reads and writes record data in a variety of formats.

The attributes of a RECORD\_ADAPTER describe where the data is located, the format, and various aspects of processing.

### DTD

```
<!ELEMENT RECORD_ADAPTER
  ( COMMENT?
  , RECORD_SOURCE?
  , DIMENSION_SOURCE?
  , RECORD_INDEX?
  , RECORD_GROUP?
  , TRANSFORMER?
  , PASS_THROUGH*
  , PROP_DECL*
  )
>
<!ATTLIST RECORD_ADAPTER
  NAME                CDATA                #REQUIRED
  FRC_PVAL_IDX        ( TRUE | FALSE )    "TRUE "
  DIRECTION           ( INPUT | OUTPUT )  #REQUIRED
  URL                 CDATA                #IMPLIED
  STATE               ( TRUE | FALSE )    #IMPLIED
  PREFIX              CDATA                #IMPLIED
  COL_DELIMITER       CDATA                #IMPLIED
  ROW_DELIMITER       CDATA                #IMPLIED
  REC_DELIMITER       CDATA                #IMPLIED
  FORMAT              CDATA                #REQUIRED
  ENCODING            CDATA                #IMPLIED
  REQUIRE_DATA        ( TRUE | FALSE )    #IMPLIED
  FILTER_EMPTY_PROPS ( TRUE | FALSE )    #IMPLIED
  MULTI               ( TRUE | FALSE )    #IMPLIED
  MULTI_PROP_NAME     CDATA                #IMPLIED
  JAVA_HOME           CDATA                #IMPLIED
  JAVA_CLASSPATH      CDATA                #IMPLIED
  JAVA_CLASSNAME      CDATA                #IMPLIED
  %compression;
```

### Attributes

The following sections describe the RECORD\_ADAPTER element's attributes.

#### NAME

Identifies the RECORD\_ADAPTER so that it can be referenced as a record source by other components in the pipeline.

#### FRC\_PVAL\_IDX

Forces Forge to create a record or property index for better performance during processing. TRUE always creates an index during processing. FALSE allows Forge to determine when an index would improve performance and to create it at that time.

#### DIRECTION

Indicates either INPUT or OUTPUT depending on whether the adapter is reading records (input) or writing records (output). If FORMAT is set to ODBC, then DIRECTION must be INPUT. If DIRECTION is set to output, then FORMAT must be either BINARY or XML.

### URL

Specifies the location of the source data. The path is either an absolute path or a path relative to the location of the Pipeline.epx file. With an absolute path, the protocol must be specified in RFC 2396 syntax. Usually this means the prefix `file:///` precedes the path to the data file. Relative URLs must not specify the protocol and are relative to the URL used to locate the Pipeline.epx file.

### STATE

Specifies the filename prefix to add when Forge writes its output files. The default is "out".

### COL\_DELIMITER

Indicates the character used to separate columns when using the DELIMITED or VERTICAL formats.

### ROW\_DELIMITER

Indicates the character used to separate rows when using the DELIMITED or VERTICAL formats.

### REC\_DELIMITER

Indicates the character or string used to separate records when using the VERTICAL format.

### FORMAT

Specifies the way in which the source data is formatted. Valid values are as follows:

- BINARY - A proprietary format.
- DELIMITED - If you input delimited data, the COL\_DELIMITER, and ROW\_DELIMITER attributes are required for further configuration.
- FIXED-WIDTH - If you input fixed-width data, the PASS\_THROUGH sub-element is required for further configuration.
- ODBC - Indicates that source data being read is a Windows ODBC datasource. This FORMAT requires two PASS\_THROUGH sub-elements: one to specify the SQL that queries the ODBC datasource and another to identify the datasource name (DSN). See the Example section in PASS\_THROUGH for a typical configuration. For information about which data types the ODBC and JDBC record adapters can fetch, see the table in the "JDBC and ODBC supported data types" section below.
- JDBC - Indicates that source data being read is a JDBC database. When using this format, you must specify a value for the JAVA\_HOME and JAVA\_CLASSPATH attributes. In addition, you must create PASS\_THROUGH sub-elements that specify the JDBC driver, the database URL, and SQL to execute against the database. If necessary, you can create PASS\_THROUGH sub-elements that specify user name and password parameters. See the Example section for PASS\_THROUGH. For information about which data types the ODBC and JDBC record adapters can fetch, see the table in the "JDBC and ODBC supported data types" section below.
- JAVA\_ADAPTER - Indicates that the data source being read is a custom record adapter that a developer created using the Endeca Record Adapter Developer Kit. When using this format, you must specify values for the JAVA\_CLASSPATH, JAVA\_CLASSNAME, and JAVA\_HOME attributes. Depending on what other parameters the adapter developer decided to expose, there may be additional requirements specified in PASS\_THROUGH sub-elements. See the *Endeca Content Adapter Developer's Guide* for more information.
- EXCHANGE - Indicates that the data source being read is a Microsoft Exchange server. When using this format, you must specify a value for the JAVA\_HOME attribute and create PASS\_THROUGH sub-elements that specify an HTTP path to the Exchange server and if authentication is required, specify a path to a `key_ring.xml` file.
- VERTICAL - Vertical data is a format that lists one property name and one property value per line, with records separated by a specified string on a line by itself. If you input vertical data, use the COL\_DELIMITER

attribute to separate property names from property values. Use the `ROW_DELIMITER` to identify the string used to separate lines; and use the `REC_DELIMITER` attribute to identify the string used to mark the end of a record.

- XML - If you input XML source data, the encoding value should be specified in the first line of the XML file.

## ENCODING

Specifies the character encoding of the source data. When you provide source data, Forge requires the correct encoding value to read the data correctly. If you do not specify encoding, Forge defaults to Latin-1. The XML format has a standard for specifying the encoding in the first line of XML file. An `ENCODING` value is ignored for OUTPUT adapters and for the XML format.

When Forge crawls documents, it detects the encoding and stores an encoding value in a property on the record named `Endeca.Document.Encoding`. You do not need to specify an `ENCODING` value when crawling documents with a SPIDER.

Forge supports all encodings supported by International Components for Unicode (ICU). There are several hundred supported `ENCODING` values and more than 1000 aliases to those encoding values, all of which are valid input to the `ENCODING` attribute. Here are a few common examples of `ENCODING` values:

- ISO-8859-1 (Latin-1)
- ISO-8859-15 (Latin-9)
- WINDOWS-1252 (CP1252, WINDOWS-1252)
- ASCII (US-ASCII)
- UTF-8
- UTF-16
- Big5

The `ENCODING` attribute may be overridden when running Forge on the command-line using the `--input-encoding` flag. This flag overrides all encodings for all input adapters (except when the XML format is being used; see below). Both the `ENCODING` attribute and the `--input-encoding` flag are ignored for OUTPUT adapters and for the XML format.

## REQUIRE\_DATA

Set to `TRUE` by default, this attribute causes Forge to generate an error if the URL does not exist or is empty. The attribute needs to be declared within an adapter only if it is being set to `FALSE` (to turn the error off).

## FILTER\_EMPTY\_PROPS

Set to `FALSE` by default, this attribute ensures that empty property values are not assigned to records. For example, if a record has no value for the "Colors" property, by default the record adapter assigns the property a value of "" (an empty string). Setting `FILTER_EMPTY_PROPS` to `TRUE` prevents this behavior. The value needs to be declared within a record adapter only if it is being set to `TRUE`.

The `FILTER_EMPTY_PROPS` attribute applies only to input record adapters (i.e., it does not apply to output record adapters). In addition, it is valid only for the Vertical, Delimited, Fixed-width, and ODBC input formats.

## MULTI

Specifies whether Forge can read data from more than one input file. The default, `FALSE`, indicates that the URL is a single input file. If set to `TRUE`, the input URL is interpreted as a pattern, and Forge reads each file matching the pattern in sequence. For example, the record adapter may specify a URL pattern of `*.update.txt`, in which case Forge reads any file in the given directory that has the `.update.txt` suffix. This value must be set to `TRUE` if you want to use `MULTI_PROP_NAME`.

When used in an OUTPUT record adapter, the `MULTI` and `MULTI_PROP_NAME` attributes specify which file each record is written to. If `MULTI="TRUE"` then Forge checks `MULTI_PROP_NAME` for the location.

## MULTI\_PROP\_NAME

Specifies a property name that indicates the location that a record coming through Forge should be written to. (Forge stores a property on each record that is the source URL for that record.) Setting this property is useful if you want to process multiple input files and also keep the output distinct. In such cases, set `MULTI="TRUE"` and then set `MULTI_PROP_NAME="propname"` to specify the location that the given record coming through Forge should be written to.

For example, suppose you have an input files named `datedfile#1.txt` and `datedfile#2.txt` that are unique based on the date in the file name. Further suppose you set `MULTI="TRUE"`, set `MULTI_PROP_NAME="Absolute_url"`, and set `OUTPUT_PREFIX="out"` in an `OUTPUT` adapter. Forge processes the files from the `Absolute_url` value of the input directory and file name, and Forge writes `out.datedfile#1.records.xml` and `out.datedfile#2.records.xml` to the `OUTPUT_URL`. The output files contain only the processed records from their corresponding input files.

### STATE

`TRUE` indicates that the value of `URL` is relative to the Forge `--stateDir` flag. This allows you to change your state directory using the `--stateDir` flag and yet not require you to modify your record adapter configuration.

### JAVA\_HOME

Specifies the location of the Java runtime engine (JRE). For example, if you have the Java SDK installed, the attribute may be set to:

```
JAVA_HOME="C:\j2dk\jre"
```

If you have only the JRE installed, the attribute may be set to:

```
JAVA_HOME="C:\j2dk"
```

This attribute is required when `FORMAT` is set to `JAVA_ADAPTER`, `JDBC`, or `EXCHANGE`.

### JAVA\_CLASSPATH

When `FORMAT="JAVA_ADAPTER"`, this attribute specifies the absolute path to the custom record adapter's `.jar` file on every machine on which Forge runs, as in this example:

```
JAVA_CLASSPATH="C:\Projects\lib\MyAdapter.jar"
```

The `.jar` file must contain the adapter class and all other classes it depends on.

When `FORMAT="JDBC"`, this attribute specifies the location of the `.jar` file that contains the JDBC driver, as in this example:

```
JAVA_CLASSPATH="/usr/local/projects/lib/oracle8i.zip"
```

If the JDBC driver you are using is distributed with the JVM, you can omit this setting (for example, the `jdbc-odbc` bridge driver does not require a classpath).

This attribute is required when `FORMAT` is set to `JAVA_ADAPTER`.

### JAVA\_CLASSNAME

Specifies the name of the adapter class to load within the `.jar` file indicated by `JAVA_CLASSPATH`, as in this example:

```
JAVA_CLASSNAME="MyAdapter"
```

This attribute is required when `FORMAT` is set to `JAVA_ADAPTER`.

### COMPRESSION\_LEVEL

Controls the level of data compression when an adapter is configured for `OUTPUT`. The default compression level for this attribute is 0 (no compression). The value `"-1"` for this attribute has been deprecated. Values 1-9 indicate increasing levels of compression.

## Sub-elements

The following table provides a brief overview of the RECORD\_ADAPTER sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <code>&lt;!-- ... --&gt;</code> .
RECORD_SOURCE	Specifies the name of a pipeline component from which this component should read records.
DIMENSION_SOURCE	Specifies the name of a pipeline component from which the component should read dimensions.
RECORD_INDEX	Specifies the property and dimension keys on which record sorts are validated. Join operations also find records that share a common key based on the RECORD_INDEX element.
RECORD_GROUP	Provides property and dimension keys for record grouping.
TRANSFORMER	Transforms source XML data into Endeca-compatible XML records according to a specified XSLT style sheet.
PASS_THROUGH	Provides additional parameters for the Java class.

## JDBC and ODBC supported data types

The following table provides information about supported data types for record adapters using either the JDBC or ODBC formats.

Data type	Supported?
CHAR	Yes, up to 1M bytes. Note that if a field in the result set is larger than 1MB (1,048,576 bytes), the result is truncated at that length.
VARCHAR	Yes, up to 1M bytes.
LONGVARCHAR (CLOB)	No
WCHAR	Yes, up to 1M bytes.
VARWCHAR	Yes, up to 1M bytes.
LONGWVARCHAR	No
DECIMAL	Yes
NUMERIC	Yes
SMALLINT	Yes
INTEGER	Yes
REAL	Yes
FLOAT	Yes
DOUBLE	Yes
BIT	Yes
TINYBIT	Yes

Data type	Supported?
BIGINT	No
BINARY	No
VARBINARY	No
LONGVARBINARY (BLOB)	No
DATE	Yes
TIME	Yes
TIMESTAMP	Yes
UTCDATETIME	No
UTCTIME	No
INTERVAL MONTH	No
INTERVAL YEAR	No
INTERVAL YEAR TO MONTH	No
INTERVAL DAY	No
INTERVAL HOUR	No
INTERVAL MINUTE	No
INTERVAL SECOND	No
INTERVAL DAY TO HOUR	No
INTERVAL DAY TO MINUTE	No
INTERVAL DAY TO SECOND	No
INTERVAL HOUR TO MINUTE	No
INTERVAL HOUR TO SECOND	No
INTERVAL MINUTE TO SECOND	No
GUID	Yes

### Example

This example shows an input record adapter reading pipe-delimited text.

```
<RECORD_ADAPTER COL_DELIMITER="|" DIRECTION="INPUT"
  FILTER_EMPTY_PROPS="TRUE" FORMAT="DELIMITED" FRC_PVAL_IDX="TRUE"
  NAME="LoadMainData" PREFIX="" REC_DELIMITER="" ROW_DELIMITER="|\n"
  URL="../incoming/wine_data.txt.gz">
  <COMMENT></COMMENT>
</RECORD_ADAPTER>
```

## RECORD\_ASSEMBLER

A RECORD\_ASSEMBLER element assembles (joins) records from multiple sources together based on a specified index.

For more information about joining source data, see the *Endeca Forge Guide*.

## DTD

```
<!ELEMENT RECORD_ASSEMBLER
  ( COMMENT?
    , RECORD_SOURCE+
    , DIMENSION_SOURCE?
    , RECORD_INDEX?
    , RECORD_JOIN
  )
>
<!ATTLIST RECORD_ASSEMBLER
  NAME          CDATA          #REQUIRED
  FRC_PVAL_IDX  ( TRUE | FALSE ) "TRUE "
```

## Attributes

The following sections describe the RECORD\_ASSEMBLER element's attributes

### NAME

Identifies the RECORD\_ASSEMBLER so that it can be referenced as a record source by other pipeline components

### FRC\_PVAL\_IDX

Forces Forge to create a record or property index for better performance during processing. TRUE always creates an index during processing. FALSE allows Forge to determine when an index would improve performance and to create it at that time.

## Sub-elements

The following table provides a brief overview of the RECORD\_ASSEMBLER sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.
RECORD_SOURCE	Specifies the name of a pipeline component from which this component should read records.
DIMENSION_SOURCE	Specifies the name of a pipeline component from which the component should read dimensions.
RECORD_INDEX	Specifies a common key used to identify records during join operations. RECORD_INDEX also specifies the property and dimension keys on which record sorts are validated.
RECORD_JOIN	Describes how to perform a join.

## Example

This example shows three sources being joined in a left join according to SID and ID keys.

```
<RECORD_ASSEMBLER NAME="ra-1" FRC_PVAL_IDX="TRUE">
  <RECORD_SOURCE>rs-1</RECORD_SOURCE>
  <RECORD_SOURCE>rs-2</RECORD_SOURCE>
```

```

<RECORD_SOURCE>rs-3</RECORD_SOURCE>
<RECORD_JOIN JOIN_TYPE="LEFT_JOIN">
  <JOIN_ENTRY>
    <RECORD_SOURCE>rs-1</RECORD_SOURCE>
    <KEY_DIMENSION TYPE="PVAL" ID="sid"/>
  </JOIN_ENTRY>
  <JOIN_ENTRY>
    <RECORD_SOURCE>rs-2</RECORD_SOURCE>
    <KEY_DIMENSION TYPE="PVAL" ID="id"/>
  </JOIN_ENTRY>
  <JOIN_ENTRY>
    <RECORD_SOURCE>rs-3</RECORD_SOURCE>
    <KEY_DIMENSION TYPE="PVAL" ID="id"/>
  </JOIN_ENTRY>
</RECORD_JOIN>
</RECORD_ASSEMBLER>

```

## RECORD\_CACHE

A RECORD\_CACHE element sorts records according to a specified RECORD\_INDEX key and loads the records in a cache for join operations.

The cache itself consists of a combination of a system's available memory and a portion of its hard drive space. If Forge runs out of available memory for the cache, Forge allocates hard drive space as necessary to continue processing.

### DTD

```

<!ELEMENT RECORD_CACHE
  ( COMMENT?
  , RECORD_SOURCE
  , DIMENSION_SOURCE?
  , RECORD_INDEX
  )
>
<!ATTLIST RECORD_CACHE
  NAME                CDATA                #REQUIRED
  FRC_PVAL_IDX        ( TRUE | FALSE )        "FALSE "
  DISK_BACKED         ( NONE | IN_MEMORY_INDEX ) "NONE "
  COMBINE_RECORDS     ( TRUE | FALSE )        #IMPLIED
  MAX_RECORDS         CDATA                #IMPLIED
>

```

### Attributes

The following sections describe the RECORD\_CACHE element's attributes.

#### NAME

A unique name for the sort operation.

#### FRC\_PVAL\_IDX

Forces Forge to create a record or property index for better performance during processing. TRUE always creates an index during processing. FALSE allows Forge to determine when an index would improve performance and to create it at that time.

#### DISK\_BACKED

This attribute is deprecated. Forge ignores any specified value.

## COMBINE\_RECORDS

Specifies whether to combine records from a single source if the records have the same key. During joins RECORD\_CACHE combines records with the same RECORD\_INDEX key. FALSE disables combining records. For more information about joining source data, see the *Endeca Forge Guide*.

## MAX\_RECORDS

This value specifies the how many records to load into the cache. Omitting this attribute loads all records.



**Note:** Use this attribute cautiously or you may get unexpected results during joins. For example, if you load only a small number of records, because you want faster processing in a testing environment, the record set may not be complete after sorting and joining. Therefore, some records may display incomplete results when navigated.

## Sub-elements

The following table provides a brief overview of the RECORD\_CACHE sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <code>&lt;!-- ... --&gt;</code> .
RECORD_SOURCE	Specifies the name of a pipeline component from which this component should read records.
DIMENSION_SOURCE	Specifies the name of a pipeline component from which the component should read dimensions.
RECORD_INDEX	Specifies the property and dimension keys on which record sorts are validated. Join operations also find records that share a common key based on the RECORD_INDEX element.

## Example

This example loads 50,000 records based on the "ID" key.

```
<RECORD_CACHE COMBINE_RECORDS="TRUE" DISK_BACKED="NONE"
  FRC_PVAL_IDX="FALSE" MAX_RECORDS="50000" NAME="record-sort">
  <RECORD_SOURCE>LoadData</RECORD_SOURCE>
  <RECORD_INDEX UNIQUE="TRUE">
    <KEY_DIMENSION ID="ID" MIN_VALS="0" TYPE="PVAL"/>
  </RECORD_INDEX>
</RECORD_CACHE>
```

## RECORD\_JOIN

A RECORD\_JOIN element describes how to perform a join contained in a RECORD\_ASSEMBLER element. For more information about joining source data, see the *Endeca Forge Guide*.

## DTD

```
<!ELEMENT RECORD_JOIN
  ( CONFIG*
  , JOIN_ENTRY+
```

```

    , WORK_ENTRY?
  )
>
<!ATTLIST RECORD_JOIN
  JOIN_TYPE          CDATA #REQUIRED
>

```

### Attributes

The following section describes the RECORD\_JOIN element's attribute.

#### JOIN\_TYPE

Indicates the type of join you want the Data Foundry to perform. Valid values for JOIN\_TYPE are: LEFT\_JOIN, INNER\_JOIN, COMBINE, OUTER\_JOIN, SWITCH, SORT\_SWITCH, DISJUNCT, FIRST\_RECORD, and HISTORY\_JOIN. See the *Endeca Forge Guide* for more detailed explanations about each type.

### Sub-elements

The following table provides a brief overview of the RECORD\_JOIN sub-elements.

Sub-element	Brief description
CONFIG	Provides generic parameterized configuration information.
RECORD_JOIN	Specifies the key used to join records.

### Example

This example shows a snippet of three record sources being joined in a left join according to SID and ID keys.

```

<RECORD_JOIN JOIN_TYPE="LEFT_JOIN">
  <CONFIG NAME="UNIQUE_RECORD_PROPS" VALUE="FALSE" />
  <JOIN_ENTRY>
    <RECORD_SOURCE>rs-1</RECORD_SOURCE>
    <KEY_DIMENSION TYPE="PVAL" ID="sid" />
  </JOIN_ENTRY>
  <JOIN_ENTRY>
    <RECORD_SOURCE>rs-2</RECORD_SOURCE>
    <KEY_DIMENSION TYPE="PVAL" ID="id" />
  </JOIN_ENTRY>
  <JOIN_ENTRY>
    <RECORD_SOURCE>rs-3</RECORD_SOURCE>
    <KEY_DIMENSION TYPE="PVAL" ID="id" />
  </JOIN_ENTRY>
</RECORD_JOIN>

```

## RECORD\_MANIPULATOR

A RECORD\_MANIPULATOR element contains EXPRESSION elements, which are evaluated against each record as Forge process it.

When an EXPRESSION is evaluated, it may modify the current record. The changes take a variety of forms depending on the expression, from adjusting property values to creating new data.



**Note:** Oracle recommends that you perform record manipulation with the PERL\_MANIPULATOR element instead of the EXPRESSION and RECORD\_MANIPULATOR elements.

## DTD

```

<!ELEMENT RECORD_MANIPULATOR
  ( COMMENT?
  , RECORD_SOURCE
  , DIMENSION_SOURCE?
  , RECORD_INDEX?
  , PROP_DECL*
  , EXPRESSION*
  )
>
<!ATTLIST RECORD_MANIPULATOR
  NAME          CDATA          #REQUIRED
  FRC_PVAL_IDX  (TRUE | FALSE)  "TRUE"
>

```

## Attributes

The following sections describe the RECORD\_MANIPULATOR element's attributes.

### NAME

Identifies the RECORD\_MANIPULATOR so that it can be referenced by other components in Pipeline.epx.

### FRC\_PVAL\_IDX

Forces Forge to create a record or property index for better performance during processing. TRUE always creates an index during processing. FALSE allows Forge to determine when an index would improve performance and to create it at that time.

## Sub-elements

The following table provides a brief overview of the RECORD\_MANIPULATOR sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.
RECORD_SOURCE	Specifies the name of a pipeline component from which this component should read records.
DIMENSION_SOURCE	Specifies the name of a pipeline component from which the component should read dimensions.
RECORD_INDEX	Specifies the property and dimension keys on which record sorts are validated. Join operations also find records that share a common key based on the RECORD_INDEX element.
EXPRESSION	Instructs Forge about how to modify data in the pipeline.

## RECORD\_SOURCE

A RECORD\_SOURCE element specifies the name of a pipeline component from which a component should read records.

Using the name reference, RECORD\_SOURCE connects components that act on records being processed (for example, RECORD\_ADAPTER, RECORD\_MANIPULATOR, JAVA\_MANIPULATOR, and so on).

**DTD**

```
<!ELEMENT RECORD_SOURCE (#PCDATA)>
```

**Attributes**

The RECORD\_SOURCE element has no attributes.

**Sub-elements**

The RECORD\_SOURCE element has no sub-elements.

**Example**

This example shows an indexer adapter that indexes source records from the RecordManipulator Adapter.

```
<INDEXER_ADAPTER NAME="IndexerAdapter"
  OUTPUT_URL=" ../partition0/forge_output/"
  FRC_PVAL_IDX="TRUE" OUTPUT_PREFIX="wine"
  OUTPUT_RECORD_FORMAT="BINARY"
  OUTPUT_DIMENSION_FORMAT="INTERNAL_DIM_XML">
  <RECORD_SOURCE>RecordManipulator</RECORD_SOURCE>
  <DIMENSION_SOURCE>DimensionServer</DIMENSION_SOURCE>
</INDEXER_ADAPTER>
```

**ROLLOVER**

A ROLLOVER element controls the point at which Forge rolls over record output to a new file.

For example, when Forge writes a large amount of records to a single file, you can set TYPE, CUTOFF, and ROLL\_URL to create another file for additional record storage.

ROLLOVER also controls which Dgidx (indexer) processes records. In cases where multiple Forge instances process records, you can control which records are sent to a particular Dgidx instance with the NUM\_IDX, PROP\_NAME, PROP\_TYPE, REMOVE\_PROP, and VALIDATE attributes. This capability is useful when you want to send records to a specific Dgidx based on a particular PROP\_NAME.

**DTD**

```
<!ELEMENT ROLLOVER EMPTY>
<!ATTLIST ROLLOVER
  NAME                CDATA                #REQUIRED
  NUM_IDX             CDATA                #IMPLIED
  PROP_NAME           CDATA                #IMPLIED
  PROP_TYPE           (ALPHA | INTEGER)     "ALPHA"
  REMOVE_PROP         (TRUE | FALSE)       "FALSE"
  VALIDATE            (TRUE | FALSE)       "FALSE"
  TYPE                (SIZE | ENTRIES)     #IMPLIED
  CUTOFF              CDATA                #IMPLIED
  ROLL_URL            CDATA                #IMPLIED
>
```

**Attributes**

The following sections describe the ROLLOVER element's attributes.

**NAME**

Identifies the type of the ROLLOVER element. NAME should be set to NAME="RECORD".

**NUM\_IDX**

Identifies the number of Dgidx instances available to Forge. This number corresponds to the number of Dgraphs, which in turn corresponds to the number of file sets Forge creates. For example, if NUM\_IDX="3" then there are three Dgidx instances, each with its own file set for record processing and its own Dgraph accessing that file set.

**PROP\_NAME**

Indicates the name of the property by which Forge assigns records to a Dgidx instance defined in NUM\_IDX. For example, if you provide PROP\_NAME="SSN", Forge assigns all records with the SSN property to one Dgidx, and consequently those records are accessed by a single Dgraph.

**PROP\_TYPE**

Indicates the property type of PROP\_NAME. Valid values are ALPHA and INTEGER. The default is ALPHA. Use INTEGER if you know exactly which Dgidx you want to process records. Use ALPHA if you are employing roll-up capabilities in your application's user interface and then send the correct roll-up records to the appropriate Dgidx.

**REMOVE\_PROP**

Removes the PROP\_NAME used to determine record processing. The default value is FALSE.

**VALIDATE**

Validates that all records are checked for the presence of multiple values of PROP\_NAME. A warning is logged for each record with multiple values.

**TYPE**

Identifies the type of measure for the CUTOFF attribute. The valid values are SIZE (which is measured in bytes) or ENTRIES (which is an integer).

**CUTOFF**

Describes the cutoff threshold after which Forge generates a new file. This value is expressed as either the number of bytes (the SIZE attribute) or an integer representing the number of record entries (the ENTRIES attribute).

When creating a new file once the CUTOFF threshold has been reached, Forge generates filenames that follow this format:

```
{prefix}[-part#][-segment#].records.{format} [.gz]
```

where each section is created only as needed. The `-part#` is used for indexer and `-segment#` is used for size, so the same part number and different segment numbers indicates the data is part of the same file set.

**ROLL\_URL**

This attribute is deprecated. The new output file is created in the directory specified by the adapter's OUTPUT\_URL attribute. The new file name conforms to the syntax listed above in the CUTOFF attribute.

**Sub-elements**

The ROLLOVER element has no sub-elements.

## Example

This example shows a ROLLOVER element that creates a new file when the number of records processed exceeds 20,000 records (based on the ENTRIES attribute). Records are assigned to one of three indexes based on the PROP1 property.

```
<ROLLOVER CUTOFF="20000" NAME="RECORD" NUM_IDX="3"
  PROP_NAME="PROP1" PROP_TYPE="ALPHA" REMOVE_PROP="FALSE"
  ROLL_URL="" TYPE="ENTRIES" VALIDATE="FALSE" />
```

## TRANSFORMER

A TRANSFORMER element instructs Forge to transform .xml from an external source into Endeca-compatible .xml according to a specified .xsl style sheet. The TRANSFORMER element may be used as a sub-element of either DIMENSION\_ADAPTER or RECORD\_ADAPTER. Therefore it can be used to transform either external dimensions or external source data into Endeca-compatible .xml for dimensions or records.

When included within a DIMENSION\_ADAPTER element, the TRANSFORMER transforms a dimension hierarchy from an external taxonomy into Endeca dimensions. Because the structure of a data hierarchy varies widely, Endeca does not provide an .xsl style sheet to transform company-specific data sets with the exception of external taxonomies from Stratify. For Stratify, Endeca provides an .xsl file. For other dimension hierarchies, you can write an .xsl file that instructs Forge how to transform your source elements into elements that conform to Endeca .xml as defined in external\_dimensions.dtd. The external\_dimensions.dtd file can be found in %ENDECA\_ROOT%\conf\dtd on Windows and \$ENDECA\_ROOT/conf/dtd on UNIX.

When included within a RECORD\_ADAPTER element, the TRANSFORMER transforms .xml source data into Endeca records. For external source data, you can write an .xsl file that instructs Forge how to transform your source elements into elements that conform to Endeca .xml as defined in records.dtd. The records.dtd file can be found in %ENDECA\_ROOT%\conf\dtd on Windows and \$ENDECA\_ROOT/conf/dtd on UNIX.

### DTD

```
<!ELEMENT TRANSFORMER EMPTY>
<!ATTLIST TRANSFORMER
  TYPE          CDATA          #REQUIRED
  URL           CDATA          #REQUIRED
>
```

### Attributes

The following section describes the TRANSFORMER element's attributes.

#### TYPE

Identifies the file format of the transformation style sheet. The only valid value is XSLT.

#### URL

Specifies the location of the .xsl file. The path is either an absolute path or a path relative to the location of the Pipeline.epx file. With an absolute path, the protocol must be specified in RFC 2396 syntax. Usually this means the prefix file:/// precedes the path to the data file. Relative URLs must not specify the protocol and are relative to the URL used to locate the Pipeline.epx file.

For a Stratify taxonomy, this URL is populated by default with the .xsl that Endeca provides.

### Sub-elements

The TRANSFORMER element has no sub-elements.

## Example

This example shows a dimension adapter reading in a Stratify taxonomy and transforming it using the `stratify2extdim.xml` stylesheet.

```
<DIMENSION_ADAPTER DIRECTION="INPUT" FORMAT="STRATIFY"
  NAME="Dimensions" URL="ask_alice_dimensions.xml">
  <TRANSFORMER TYPE="XSLT" URL="stratify2extdim.xml"/>
</DIMENSION_ADAPTER>
```

## UPDATE\_ADAPTER

As part of a partial update Pipeline.epx, an UPDATE\_ADAPTER element writes partial update information for a running Endeca MDEX Engine to perform a live update.

An update adapter is typically the last component of a partial update Pipeline.epx file. See the *Endeca Partial Updates Guide* for additional details.

### DTD

```
<!ELEMENT UPDATE_ADAPTER
  ( COMMENT?,
    RECORD_SOURCE,
    DIMENSION_SOURCE*,
    RECORD_ID_SPEC?,
    ROLLOVER?
  )
>
<!ATTLIST UPDATE_ADAPTER
  NAME                CDATA                #REQUIRED
  OUTPUT_URL          CDATA                #REQUIRED
  OUTPUT_PREFIX       CDATA                #IMPLIED
  MULTI               ( TRUE | FALSE )     #IMPLIED
  MULTI_PROP_NAME     CDATA                #IMPLIED
  FILTER_UNKNOWN_PROPS ( TRUE | FALSE )     "TRUE"
  %compression;
```

### Attributes

The following section describes the UPDATE\_ADAPTER element's attributes.

#### NAME

Identifies the UPDATE\_ADAPTER so that it can be referenced by other pipeline components.

#### OUTPUT\_URL

Identifies the directory to which Forge writes its files and processed records. The path is either an absolute path or a path relative to the location of the Pipeline.epx file. With an absolute path, the protocol must be specified in RFC 2396 syntax. Usually this means the prefix `file:///` precedes the path to the data file. Relative URLs must not specify the protocol and are relative to the URL used to locate the Pipeline.epx file.

#### OUTPUT\_PREFIX

Specifies the filename prefix to add when Forge writes its output files. The default is `out`.

#### MULTI

When used in an INPUT adapter, the MULTI attribute specifies whether Forge can read data from more than one input file. The default, FALSE, indicates that the URL is a single input file. If set to TRUE, the input URL

is interpreted as a pattern, and Forge reads each file matching the pattern in sequence. For example, the adapter may specify a URL pattern of `*.update`, in which case Forge reads any file in the given directory that has the `.update` suffix. This value must be set to `TRUE` if you want to use `MULTI_PROP_NAME`.

When used in an `OUTPUT` adapter, the `MULTI` and `MULTI_PROP_NAME` attributes specify which file each record is written to. If `MULTI="TRUE"` then Forge checks `MULTI_PROP_NAME` for the location.

### **MULTI\_PROP\_NAME**

Specifies a property name that indicates the location that a record coming through Forge should be written to. (Forge stores a property on each record that is the source URL for that record.) Setting this property is useful if you want to process multiple input files and also keep the output distinct. In such cases, set `MULTI="TRUE"` and then set `MULTI_PROP_NAME="propname"` to specify the location that the given record coming through Forge should be written to.

For example, suppose you have an input files named `datedfile#1.txt` and `datedfile#2.txt` that are unique based on the date in the file name. Further suppose you set `MULTI="TRUE"`, set `MULTI_PROP_NAME="Absolute_url"`, and set `OUTPUT_PREFIX="out"` in an `OUTPUT` adapter. Forge processes the files from the `Absolute_url` value of the input directory and file name, and Forge writes `out.datedfile#1.records.xml` and `out.datedfile#2.records.xml` to the `OUTPUT_URL`. The output files contain only the processed records from their corresponding input files.

### **FILTER\_UNKNOWN\_PROPS**

Specifies whether unmapped properties should be removed from the processed records. The default value is `TRUE`. The value in a partial update pipeline should be the same as in the baseline pipeline.

### **COMPRESSION\_LEVEL**

Controls the level of data compression when writing data. The default compression level for this attribute is 0 (no compression). The value `"-1"` for this attribute has been deprecated. Values 1-9 indicate increasing levels of compression.

### **Sub-elements**

The following table provides a brief overview of the `UPDATE_ADAPTER` sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <code>&lt;!-- ... --&gt;</code> .
RECORD_SOURCE	Specifies the name of a pipeline component from which this component should read records.
DIMENSION_SOURCE	Specifies the name of a pipeline component from which the component should read dimensions.
ROLLOVER	Controls the point at which record output gets rolled over to a new file, and which Dgidx (indexer) processes records.

### **Example**

This example shows a simple update adapter.

```
<UPDATE_ADAPTER NAME="UpdateOutput" OUTPUT_PREFIX="update"
  OUTPUT_URL=" ../data/partition0/forge_output/"
  FILTER_UNKNOWN_PROPS="TRUE">
  <RECORD_SOURCE>RecordManipulator</RECORD_SOURCE>
```

```
<DIMENSION_SOURCE>DimensionServer</DIMENSION_SOURCE>
</UPDATE_ADAPTER>
```

## Dimsearch\_config.xml elements

The Dimsearch\_config.xml file controls how dimension searches behave.

This file configures search matching, spelling correction, filtering, and relevance ranking for dimension search. These options are configured in the file's root element DIMSEARCH\_CONFIG.

### DIMSEARCH\_CONFIG

A DIMSEARCH\_CONFIG element sets up the configuration of dimensions for dimension searches. Dimension searches search against the text collection that consists of the names of all the dimension values in the data set.

#### DTD

```
<!ELEMENT DIMSEARCH_CONFIG (COMMENT?, PARTIAL_MATCH?, AUTO_SUGGEST?)>
<!ATTLIST DIMSEARCH_CONFIG
  FILTER_FOR_ANCESTORS (TRUE | FALSE) "FALSE"
  SEARCH_INERT_DVALS (TRUE | FALSE) "FALSE"
  RELRANK_STRATEGY CDATA #IMPLIED
  RETURN_RELRANK_SCORE (TRUE | FALSE) "FALSE"
  ENABLE_AUTO_SUGGEST (TRUE | FALSE) #IMPLIED
>
```

#### Attributes

The following sections describe the DIMSEARCH\_CONFIG element's attributes.

#### FILTER\_FOR\_ANCESTORS

When set to TRUE, the results of a dimension search return only the highest ancestor dimension value. This means that if both red zinfandel and red wine match a search query for "red" and FILTER\_FOR\_ANCESTORS is set to true, only the red wine dimension value is returned. When set to FALSE, then both dimension values are returned. The default value is FALSE.

#### SEARCH\_INERT\_DVALS

When set to TRUE, certain non-navigable dimension values, such as dimension roots, are also returned as the result of a dimension search query. The default value is FALSE.

#### RELRANK\_STRATEGY

Specifies the name of a relevance ranking strategy for dimension search.

#### RETURN\_RELRANK\_SCORE

Specifies that dimension search results should include the score assigned to each result by the relevance ranking system. Setting this value to TRUE is that same as running the Dgraph with the --stat-rel flag. The default value is FALSE.

#### ENABLE\_AUTO\_SUGGEST

Specifies whether automatic spelling correction is enabled for dimension searches. TRUE enables automatic spelling correction. FALSE disables it.

## Sub-elements

The following table provides a brief overview of the DIMSEARCH\_CONFIG sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.
PARTIAL_MATCH	Specifies if partial query matches should be supported for the dimension.
AUTO_SUGGEST	Specifies how to configure automatic spelling correction for either record searches or dimension searches.

## Example

This example shows a configuration that displays ancestor dimension values but disables auto suggest recommendations.

```
<DIMSEARCH_CONFIG FILTER_FOR_ANCESTORS="FALSE" SEARCH_INERT_DVALS="FALSE"
  ENABLE_AUTO_SUGGEST="FALSE" />
```

# Precedence\_rules.xml elements

The precedence\_rules.xml file contains the precedence rules for your application.

Precedence rules allow your application to delay dimension display until the user triggers the display. In other words, precedence rules are triggers that cause dimensions that were not previously displayed to now be available. This makes navigation through the data easier, and is essential to avoid information overload problems.

## PRECEDENCE\_RULE

The PRECEDENCE\_RULE element allows your application to delay dimension display until the user triggers the display.

Precedence rules are triggers that cause dimensions that were not previously displayed to now be available. This makes navigation through the data easier and is essential to avoid information overload problems.

For example, one might not want both the "Country" and "State" dimensions to appear simultaneously in a geographical data set. A PRECEDENCE\_RULE could be defined so that the "State" dimension appears only after a dimension value from the "Country" dimension is selected. In this example, Country would correspond to the source dimension value and State would correspond to the destination dimension value.

## DTD

```
<!ELEMENT PRECEDENCE_RULE EMPTY>
<!ATTLIST PRECEDENCE_RULE
  TYPE          ( STANDARD | LEAF )      #REQUIRED
  SRC_DVAL_ID   CDATA                   #REQUIRED
  DEST_DVAL_ID  CDATA                   #REQUIRED
>
```

**Attributes**

The following section describes the PRECEDENCE\_RULE element's attributes.

**TYPE**

The type of source dimension value for a PRECEDENCE\_RULE. There are two possible values: STANDARD and 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).

**SRC\_DVAL\_ID**

Specifies the source dimension value that must be selected before the user can see the destination dimension value (DEST\_DVAL\_ID).

**DEST\_DVAL\_ID**

Specifies the destination dimension value that appears after the source dimension value is selected.

**Sub-elements**

PRECEDENCE\_RULE contains no sub-elements.

**Example**

This example shows a STANDARD-type precedence rule taken from the wine reference.

```
<PRECEDENCE_RULES>
  <PRECEDENCE_RULE DEST_DVAL_ID="11" SRC_DVAL_ID="8" TYPE="STANDARD" />
</PRECEDENCE_RULES>
```

**PRECEDENCE\_RULES**

A PRECEDENCE\_RULES element specifies the individual precedences rules available to your pipeline. Each precedence rule is represented by an individual PRECEDENCE\_RULE element.

**DTD**

```
<!ELEMENT PRECEDENCE_RULES
  ( COMMENT?
    , PRECEDENCE_RULE*
  )
>
```

**Attributes**

The PRECEDENCE\_RULES element has no attributes.

**Sub-elements**

The following table provides a brief overview of the PRECEDENCE\_RULES sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.
PRECEDENCE_RULE	Allows your application to delay dimension display until the user triggers the display.

### Example

This example shows a STANDARD-type precedence rule taken from the wine reference.

```
<PRECEDENCE_RULES>
  <PRECEDENCE_RULE DEST_DVAL_ID="11" SRC_DVAL_ID="8" TYPE="STANDARD" />
</PRECEDENCE_RULES>
```

## Profiles.xml elements

The Profiles.xml file contains a PROFILE element for every user profile in your project.

PROFILE elements are grouped in the PROFILES root element.

## PROFILES

A PROFILES element contains any number of PROFILE elements to indicate a user type or user group.

Each profile is represented by an individual PROFILE element, which is an element in the Common.dtd file.

### DTD

```
<!ELEMENT PROFILES
  ( COMMENT?
    , PROFILE*
  )
>
```

### Attributes

The PROFILES element has no attributes.

### Sub-elements

The following table provides a brief overview of the PROFILES sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.
PROFILE	Specifies a user type or user group. Profiles are used as a type of dynamic business rule trigger.

**Example**

This example identifies a user type named premium\_subscriber.

```
<PROFILES>
  <PROFILE>premium_subscriber</PROFILE>
</PROFILES>
```

## Project.xml elements

The Project.xml file points to all of the files that are input to Forge and Dgidx.

The project files include dimensions.xml and the index configuration files. You do not typically need to edit this file.

## DIMENSION\_INPUTS

The DIMENSION\_INPUTS element identifies the dimensions.xml file used to define the project.

**DTD**

```
<!ELEMENT DIMENSION_INPUTS
  ( INPUT+
  )
>
```

**Attributes**

The DIMENSION\_INPUTS element has no attributes.

**Sub-elements**

The following table provides a brief overview of the RECORD\_INPUTS sub-element.

Sub-element	Brief description
INPUT	Provides details about the specified file.

**Example**

The following example defines a wine project.

```
<PROJECT NAME="wine">
  <DIMENSION_INPUTS>
    <INPUT FORMAT="XML" URL="wine.dimensions.xml"/>
  </DIMENSION_INPUTS>
  <RECORD_INPUTS>
    <INPUT FORMAT="BINARY" NUM_PARTITIONS="1" PARTITION="0"
      SEGMENT="0" URL="wine.records.binary"/>
  </RECORD_INPUTS>
</PROJECT>
```

## INPUT

The INPUT element provides specific information about files listed in the PROJECT element.

**DTD**

```

<!ELEMENT INPUT EMPTY>
<!ATTLIST INPUT
  FORMAT          CDATA      #REQUIRED
  URL             CDATA      #REQUIRED
  NUM_PARTITIONS  CDATA      #REQUIRED
  PARTITION       CDATA      #IMPLIED
  SEGMENT         CDATA      #IMPLIED
>

```

**Attributes**

The following section describes the INPUT element's attributes.

**FORMAT**

The format of the data in the file. Depending on the file type, the value can be XML or BINARY.

**URL**

Path to the file, with a prefix matching the project name.

**NUM\_PARTITIONS**

For RECORD\_INPUTS, the number of partitions.

**PARTITION**

For RECORD\_INPUTS, specifies the location of a portion of the data. This attribute is used when there are multiple Dgraphs.

**SEGMENT**

For RECORD\_INPUTS, when rollover is turned TRUE, each partition may be further divided into multiple segments. A SEGMENT element specifies the location of one of these segments.

**Sub-elements**

The INPUT element has no sub-elements.

**Example**

The following example defines a wine project.

```

<PROJECT NAME="wine">
  <DIMENSION_INPUTS>
    <INPUT FORMAT="XML" URL="wine.dimensions.xml" />
  </DIMENSION_INPUTS>
  <RECORD_INPUTS>
    <INPUT FORMAT="BINARY" NUM_PARTITIONS="1" PARTITION="0"
      SEGMENT="0" URL="wine.records.binary" />
  </RECORD_INPUTS>
</PROJECT>

```

**PROJECT**

The PROJECT element defines all of the elements in a project.

**DTD**

```

<!ELEMENT PROJECT
  ( DIMENSION_INPUTS?
    , RECORD_INPUTS?
  )
>
<!ATTLIST PROJECT
  NAME      CDATA      #REQUIRED
>

```

**Attributes**

The following section describes the PROJECT element's attribute.

**NAME**

A unique name for this project.

**Sub-elements**

The following table provides a brief overview of the PROJECT sub-elements.

Sub-element	Brief description
DIMENSION_INPUTS	Path to the dimensions.xml file for the project.
RECORD_INPUTS	Path to and information about the record source for the project.

**Example**

The following example defines a wine project.

```

<PROJECT NAME="wine">
  <DIMENSION_INPUTS>
    <INPUT FORMAT="XML" URL="wine.dimensions.xml" />
  </DIMENSION_INPUTS>
  <RECORD_INPUTS>
    <INPUT FORMAT="BINARY" NUM_PARTITIONS="1" PARTITION="0"
      SEGMENT="0" URL="wine.records.binary" />
  </RECORD_INPUTS>
</PROJECT>

```

**RECORD\_INPUTS**

The RECORD\_INPUTS element identifies and describes the file used to import records into the project.

**DTD**

```

<!ELEMENT RECORD_INPUTS
  ( INPUT+
  )
>

```

**Attributes**

The RECORD\_INPUTS element has no attributes.

### Sub-elements

The following table provides a brief overview of the RECORD\_INPUTS sub-element.

Sub-element	Brief description
INPUT	Provides details about the specified file.

### Example

The following example defines a wine project.

```
<PROJECT NAME="wine">
  <DIMENSION_INPUTS>
    <INPUT FORMAT="XML" URL="wine.dimensions.xml" />
  </DIMENSION_INPUTS>
  <RECORD_INPUTS>
    <INPUT FORMAT="BINARY" NUM_PARTITIONS="1" PARTITION="0"
      SEGMENT="0" URL="wine.records.binary" />
  </RECORD_INPUTS>
</PROJECT>
```

## Prop\_refs.xml elements

The Prop\_refs.xml file contains a PROP\_REF element for every property in your pipeline.

The PROP\_REF element indicates the name and data type of each property. PROP\_REF elements are grouped in the PROP\_REFS root element.

### PROP\_REF

The PROP\_REF element specifies the name and data type of properties in your pipeline.

#### DTD

```
<!ELEMENT PROP_REF EMPTY>
<!ATTLIST PROP_REF
  NAME          CDATA          #REQUIRED
  TYPE          ( ALPHA
                  | INTEGER
                  | DOUBLE
                  | FIXEDPOINT
                  | FILE_PATH
                  | GEOCODE
                  | DATETIME
                  | DURATION
                  | TIME )      #REQUIRED
>
```

#### Attributes

The following sections describe the PROP\_REF element's attributes.

##### NAME

The unique identifier for this property.

## TYPE

Indicates the data type of this property:

- ALPHA - used for alphanumeric data.
- INTEGER - used for integer data.
- DOUBLE - used for floating point data.
- FIXEDPOINT - not supported.
- FILE\_PATH - deprecated.
- GEOCODE - a pair of double precision floating point values representing a latitude and longitude pair. Used for geospatial filtering and sorting.
- DATETIME - used for values that represent a time of the day on a given date.
- DURATION - used for lengths of time; intended for use with Endeca Analytics.
- TIME - used for the time of day; intended for use with Endeca Analytics.

## Sub-elements

The PROP\_REF element has no sub-elements.

## Example

This example shows some of the properties defined in a project.

```
<PROP_REFS>
  <PROP_REF NAME="P_Body" TYPE="ALPHA" />
  <PROP_REF NAME="P_Description" TYPE="ALPHA" />
  <PROP_REF NAME="P_Price" TYPE="DOUBLE" />
  <PROP_REF NAME="P_Year" TYPE="INTEGER" />
  <PROP_REF NAME="Location" TYPE="GEOCODE" />
  ...others deleted for simplicity...
</PROP_REFS>
```

## PROP\_REFS

A PROP\_REFS element contains the properties specified for your pipeline.

Each individual property is specified in a PROP\_REF element.

## DTD

```
<!ELEMENT PROP_REFS
  ( COMMENT?
    ; PROP_REF*
  )
>
```

## Attributes

The PROP\_REFS element has no attributes.

## Sub-elements

The following table provides a brief overview of the PROP\_REFS sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <code>&lt;!-- ... --&gt;</code> .
PROP_REF	Specifies the name and data type of properties in your pipeline.

### Example

This example shows some of the properties defined in a project.

```
<PROP_REFS>
  <PROP_REF NAME="P_Body" TYPE="ALPHA" />
  <PROP_REF NAME="P_Description" TYPE="ALPHA" />
  <PROP_REF NAME="P_Price" TYPE="DOUBLE" />
  <PROP_REF NAME="P_Year" TYPE="INTEGER" />
  <PROP_REF NAME="Location" TYPE="GEOCODE" />
  ...others deleted for simplicity...
</PROP_REFS>
```

## Record\_filter.xml elements

The Record\_filter.xml file specifies the properties enabled for a custom catalog filter.

Each property is indicated by a RECORD\_FILTER element. See the "Record Filters" chapter in the *Endeca Development Guide* for details.

### RECORD\_FILTER

A RECORD\_FILTER element adds the specified property to a custom catalog filter.

Custom catalog filters present a subset of the data to the end-user. See the "Record Filters" chapter in the *Endeca MDEX Engine Development Guide* for details.

#### DTD

```
<!ELEMENT RECORD_FILTER
  ( COMMENT?
    , PROPNAME*
  )
>
```

#### Attributes

The RECORD\_FILTER element has no attributes.

#### Sub-elements

The following table provides a brief overview of the RECORD\_FILTER sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <code>&lt;!-- ... --&gt;</code> .
PROPNAME	Represents a property.

### Example

This example adds an access control list property to the records being processed.

```
<RECORD_FILTER>
  <PROPNAME>Endeca.ACL.Allow.Read</PROPNAME>
</RECORD_FILTER>
```

## Record\_id\_prop.xml elements

The Record\_id\_prop.xml file contains a RECORD\_ID\_PROP element that specifies a property to identify records during partial updates.

When implementing partial updates, the RECORD\_ID\_PROP element uses this property to preserve stable record IDs across baseline runs. That is, a record will have the same ID in the next update as in the current update.

## RECORD\_ID\_PROP

A RECORD\_ID\_PROP element indicates that this property should be used as the record ID to identify records.

If a property is to be used as a record ID, it must fulfill the following criteria:

- It must be a 32-bit unsigned integer.
- The value of this property must be unique for each record.
- No more than one property may be specified by the sub-element PROPNAME.



**Note:** This element is very similar to RECORD\_SPEC which also identifies records during updates. The difference is that the value of PROPNAME in RECORD\_ID\_PROP must be an integer; whereas, the value of PROPNAME in RECORD\_SPEC can be a string.

### DTD

```
<!ELEMENT RECORD_ID_PROP
  ( COMMENT?
    ; PROPNAME?
  )
>
```

### Attributes

The RECORD\_ID\_PROP element has no attributes.

### Sub-elements

The following table provides a brief overview of the RECORD\_ID\_PROP sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <code>&lt;!-- ... --&gt;</code> .
PROPNAME	Represents the property to be used as a record ID.

### Example

This example identifies records by their RecordIdProp property.

```
<RECORD_ID_PROP>
  <PROPNAME>RecordIdProp</PROPNAME>
</RECORD_ID_PROP>
```

## Record\_sort\_config.xml elements

The Record\_sort\_config.xml file specifies properties or dimensions that users can sort by in an application.

The root element RECORD\_SORT\_CONFIG contains a list of properties or dimensions users can sort by. Each property or dimension is specified in a RECORD\_SORT element.

### RECORD\_SORT

A RECORD\_SORT element specifies whether the MDEX Engine uses precomputed sort on this property or dimension, to optimize sort processing.

#### DTD

```
<!ELEMENT RECORD_SORT EMPTY>
<!ATTLIST RECORD_SORT
  NAME          CDATA          #REQUIRED
>
```

#### Attributes

The following sections describe the RECORD\_SORT element's attributes.

#### NAME

Specifies the name of the property or dimension on which the MDEX Engine uses precomputed sort, to optimize processing.

#### Sub-elements

The RECORD\_SORT element has no sub-elements.

#### Example

This example shows three properties that the MDEX Engine uses for precomputed sort:

```
<RECORD_SORT_CONFIG>
  <RECORD_SORT NAME="Name" />
  <RECORD_SORT NAME="P_PriceStr" />
```

```
<RECORD_SORT NAME="P_Year" />
</RECORD_SORT_CONFIG>
```

## RECORD\_SORT\_CONFIG

A RECORD\_SORT\_CONFIG element enables precomputed sort by the MDEX Engine, according to specified properties or dimensions in an application.

Within the RECORD\_SORT\_CONFIG element, each property or dimension that you want to enable for precomputed sorting should be specified in a RECORD\_SORT sub-element.

### DTD

```
<!ELEMENT RECORD_SORT_CONFIG
  ( COMMENT?
    , RECORD_SORT*
  )
>
```

### Attributes

The RECORD\_SORT\_CONFIG element has no attributes.

### Sub-elements

The following table provides a brief overview of the RECORD\_SORT\_CONFIG sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <code>&lt;!-- ... --&gt;</code> .
RECORD_SORT	Specifies a property or dimension on which the MDEX Engine uses precomputed sort, to optimize processing.

### Example

This example shows three properties in an application for which the MDEX Engine uses precomputed sort:

```
<RECORD_SORT_CONFIG>
  <RECORD_SORT NAME="Name" />
  <RECORD_SORT NAME="P_PriceStr" />
  <RECORD_SORT NAME="P_Year" />
</RECORD_SORT_CONFIG>
```

## Record\_spec.xml element

The Record\_spec.xml file contains a RECORD\_SPEC element that specifies a property to identify records during partial updates.

When implementing partial updates, the RECORD\_SPEC element uses this property to preserve stable record IDs across baseline runs. That is, a record will have the same ID in the next update as in the current update. For more information, see the *Endeca MDEX Engine Partial Updates Guide*.

## RECORD\_SPEC

A RECORD\_SPEC element specifies that this property should be used as the record specification to identify records during partial updates.

If a property is to be used as a record specification property, it must fulfill the following criteria:

- Every record must have exactly one assignment of this property.
- The value of this property must be unique for each record.
- No more than one property is identified by the PROPNAME sub-element.



**Note:** This element is very similar to RECORD\_ID\_PROP which also identifies records during partial updates. The difference is that the value of PROPNAME in RECORD\_ID\_PROP must be an integer; whereas, the value of PROPNAME in RECORD\_SPEC can be a string.

### DTD

```
<!ELEMENT RECORD_SPEC
  ( COMMENT?
    , PROPNAME?
  )
>
```

### Attributes

The RECORD\_SPEC element has no attributes.

### Sub-elements

The following table provides a brief overview of the RECORD\_SPEC sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.
PROPNAME	Specifies the property to be used as the record specification property.

### Example

This example sets the identifies records to update by the SKU\_ID property.

```
<RECORD_SPEC>
  <PROPNAME>SKU_ID</PROPNAME>
</RECORD_SPEC>
```

## Recsearch\_config.xml elements

The Recsearch\_config.xml configures record search.

In the file's RECSEARCH\_CONFIG root element, you can also configure Did You Mean and automatic spelling correction options for dimension search. For more information about spelling corrections, see "Using Spelling Correction and Did You Mean" in the *MDEX Development Guide*.

## RECSEARCH\_CONFIG

A RECSEARCH\_CONFIG element sets up the configuration of dimensions for dimension searches.

Dimension searches search against the text collection that consists of the names of all the dimension values in the data set.



**Note:** Do not modify the ENABLE\_AUTO\_SUGGEST and ENABLE\_DID\_YOU\_MEAN attributes for this element, since they are no longer supported starting with the MDEX Engine 6.1.0. To configure Did You Mean and automatic spelling correction options, you must use the corresponding flags for the Dgraph. For more information about spelling corrections, see "Using Spelling Correction and Did You Mean" in the *MDEX Engine Development Guide*.

### DTD

```
<!ELEMENT RECSEARCH_CONFIG
  ( COMMENT?
  , DID_YOU_MEAN?
  , AUTO_SUGGEST?
  , SEARCH_INTERFACE*
  )
>
<!ATTLIST RECSEARCH_CONFIG
  WHY_MATCH          (NONE | CONCISE | COMPLETE)  "NONE"
  WORD_INTERP        (TRUE | FALSE)              "FALSE"
  RETURN_RELRANK_SCORE (TRUE | FALSE)                "FALSE"
  ENABLE_DID_YOU_MEAN (TRUE | FALSE)                #IMPLIED
  ENABLE_AUTO_SUGGEST (TRUE | FALSE)                #IMPLIED
>
```

### Attributes

The following sections describe the RECSEARCH\_CONFIG element's attributes.

#### WHY\_MATCH

When set to COMPLETE, this attribute specifies whether to enable computation of "Why Did It Match" attributes returned with full text search queries. These dynamic attributes contain a copy of the property/dimension key and value that caused the match along with query interpretation notes (spelling, thesaurus, etc.).

When set to CONCISE, this attribute produces more concise dynamic attribute values containing only the property/dimension key and query interpretation notes. This is useful when a property value might include large amounts of text, such as document contents.

When set to NONE, this feature is disabled. The default value is NONE.

#### WORD\_INTERP

Specifies whether to enable word interpretation forms (see-also suggestions) of user query terms considered by the text search engine while processing record search requests. The default value is FALSE.

#### RETURN\_RELRANK\_SCORE

Specifies whether to return dimension value attributes that indicate the relevance rank score for dimension search results. The default value is FALSE.

#### ENABLE\_DID\_YOU\_MEAN



**Note:** Do not change the value for this attribute in the file. The Dgraph ignores the changed value starting with the MDEX Engine v.6.1.0. Instead, enable "Did you mean" with the `--dym` flag for the Dgraph.

Specifies whether the application should return explicit alternatives for search terms as a part of spelling correction. TRUE enables Did You Mean suggestions. FALSE disables them. Use the DID\_YOU\_MEAN sub-element to configure Did You Mean suggestions.

### ENABLE\_AUTO\_SUGGEST



**Note:** Do not change the value for this attribute in the file. The Dgraph ignores the changed value starting with the MDEX Engine v.6.1.0. Instead, enable automatic correction with the `--spl` flag for the Dgraph.

Specifies whether automatic spelling correction is enabled for record searches. TRUE enables automatic spelling correction. FALSE disables it. Use the AUTO\_SUGGEST sub-element to configure automatic spelling corrections.

### Sub-elements

The following table provides a brief overview of the RECSEARCH\_CONFIG sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <code>&lt;!-- ... --&gt;</code> .
DID_YOU_MEAN	Specifies how to configure explicit alternatives for search terms as a part of spelling correction.
AUTO_SUGGEST	Specifies how to configure automatic spelling correction for either record searches or dimension searches.
SEARCH_INTERFACE	Represents a named collection of dimensions and/or properties.

### Example

This example shows the configuration for the wine reference implementation.

```
<RECSEARCH_CONFIG>
  <SEARCH_INTERFACE CROSS_FIELD_BOUNDARY="NEVER"
    CROSS_FIELD_RELEVANCE_RANK="0"
    DEFAULT_RELRANK_STRATEGY="All" NAME="All">
    <MEMBER_NAME RELEVANCE_RANK="4">P_WineType</MEMBER_NAME>
    <MEMBER_NAME RELEVANCE_RANK="3">P_Name</MEMBER_NAME>
    <MEMBER_NAME RELEVANCE_RANK="2">P_Winery</MEMBER_NAME>
    <MEMBER_NAME RELEVANCE_RANK="1">P_Description</MEMBER_NAME>
  </SEARCH_INTERFACE>
</RECSEARCH_CONFIG>
```

## Recsearch\_indexes.xml elements

The Recsearch\_indexes.xml file enables specified properties and dimensions for record searching.

Record search finds all records in an Endeca application that have a property or dimension whose value matches a term the user provides. The RECSEARCH\_INDEX element specifies each property or dimension and indicates whether to consider ancestor dimension values when matching a dimension search query. RECSEARCH\_INDEX elements are grouped in the RECSEARCH\_INDEXES parent element.

## RECSEARCH\_INDEX

A RECSEARCH\_INDEX element enables a property or dimension for record search.

### DTD

```
<!ELEMENT RECSEARCH_INDEX (SEARCH_INDEX)>
<!ATTLIST RECSEARCH_INDEX
  NAME          CDATA          #REQUIRED
  RECSEARCH_HIERARCHY (TRUE | FALSE) "FALSE"
>
```

### Attributes

The following sections describe the RECSEARCH\_INDEX element's attributes.

#### NAME

Specifies the dimension or property name that you want to enable for record search.

#### RECSEARCH\_HIERARCHY

This attribute is used for dimensions. When set to TRUE, RECSEARCH\_HIERARCHY specifies that a record search on dimension values from this dimension should match against words in the entire set of dimension values in the hierarchy rather than match only against one dimension value. For example, if a dimension value hierarchy is Electronics > Computers > Desktop PCs, then a record search on the Electronics dimension for "desktop computers" returns records associated with Desktop PCs because "desktop" matches the leaf and "computers" matches its parent. Although the two words occur in distinct dimension values, the leaf records are returned because the entire hierarchy is searched.

When set to FALSE, RECSEARCH\_HIERARCHY specifies that a record search on dimension values from this dimension may not match against words in the entire set of dimension values in the hierarchy. For example, if a dimension value hierarchy is Electronics > Computers > Desktop PCs, then a record search on the Electronics dimension for "desktop computers" returns no records because the matches for "desktop" and "computers" span multiple dimensions in the hierarchy. The default value is FALSE.

### Sub-elements

The following table provides a brief overview of the RECSEARCH\_INDEX sub-elements.

Sub-element	Brief description
SEARCH_INDEX	Controls the construction of the search index for both record search and dimension search.

### Example

This example enables the Winery dimension for record search.

```
<RECSEARCH_INDEX NAME="Winery" RECSEARCH_HIERARCHY="TRUE">
  <SEARCH_INDEX>
    <POSITIONAL_INDEX/>
  </SEARCH_INDEX>
</RECSEARCH_INDEX>
```

## RECSEARCH\_INDEXES

A RECSEARCH\_INDEXES element lists the properties and dimensions enabled for record search.

Each property and dimension enabled for record search is specified in a RECSEARCH\_INDEX element.



**Note:** Several settings in this element are ignored starting with the MDEX Engine version 6.1.2. These settings are: MAX\_NGRAM\_LENGTH, DICTIONARY\_MAX\_NGRAM\_LENGTH, and DICTIONARY\_WILDCARD. Do not remove these settings from the file, since the DTD depends on them. Starting with the version 6.1.2, the MDEX Engine uses a simplified mechanism for wildcard search. This mechanism does not require specifying these settings. For more information on wildcard search, see the *Endeca MDEX Engine Development Guide* version 6.1.2 or higher.

## DTD

```
<!ELEMENT RECSEARCH_INDEXES
  ( COMMENT?
    , RECSEARCH_INDEX*
  )
>
<!ATTLIST RECSEARCH_INDEXES
  INMEM_INDEX_THRESHOLD          CDATA          #IMPLIED
  MAX_NGRAM_LENGTH              CDATA          #IMPLIED
  DICTIONARY_WILDCARD           ( TRUE | FALSE ) #IMPLIED
  DICTIONARY_MAX_NGRAM_LENGTH   CDATA          #IMPLIED
  MIN_WORD_OCCURRENCES         CDATA          #IMPLIED
  MIN_WORD_LENGTH               CDATA          #IMPLIED
  MAX_WORD_LENGTH               CDATA          #IMPLIED
>
```

## Attributes

The following section describes the RECSEARCH\_INDEXES element's attributes.

### INMEM\_INDEX\_THRESHOLD

*In MDEX Engine 6.4.0, this attribute was deprecated.* Specifies the maximum text size (in MB) for which text indexing will be done entirely in memory. If the text size is larger than this threshold, then temporary files on disk will be used. Typically, you do not need to modify the value of this attribute. However, there are two infrequent scenarios in which you may want to adjust the value of this attribute:

- If the machine running Dgidx has less than the recommended amount of RAM, you might reduce this value.
- If other indexing structures cause Dgidx to run out of memory, you might reduce this value to allow more memory for other purposes.

### MAX\_NGRAM\_LENGTH



**Note:** The value specified in this setting is ignored in the MDEX Engine starting with its version 6.1.2. However, do not remove this XML setting.

Specifies the maximum ngram length that should be indexed. All substrings of this length or shorter will be indexed. If a user does a wildcard search with a string that is less than or equal to this length, then exact results can be returned directly from the index. If the wildcard search includes a substring longer than this length, then the results returned from the index will be post-processed so that false positives are eliminated.

### DICTIONARY\_WILDCARD



**Note:** The value specified in this setting is ignored in the MDEX Engine starting with its version 6.1.2. However, do not remove this XML setting.

Specifies whether a wildcard index should be generated for the dictionary of words for the text collection containing this element. The default value of this attribute is TRUE when the containing element consists of off-line documents and FALSE otherwise.

### DICTIONARY\_MAX\_NGRAM\_LENGTH



**Note:** The value specified in this setting is ignored in the MDEX Engine starting with its version 6.1.2. However, do not remove this XML setting.

Specifies the maximum ngram length that should be indexed in the dictionary wildcard index.

### MIN\_WORD\_OCCURRENCES

Specifies the minimum number of times a word must occur for it to be indexed for spelling correction. The default value is 4.

### MIN\_WORD\_LENGTH

Specifies the minimum number of a characters that a word must contain for it to be indexed for spelling correction. The default value is 3 (words that are three characters or larger are added to the index).

### MAX\_WORD\_LENGTH

Specifies the maximum number of a characters that a word may contain for it to be indexed for spelling correction. The default value is 16 (words that are 16 characters or fewer are added to the index).

### Sub-elements

The following table provides a brief overview of the RECSEARCH\_INDEXES sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.
RECSEARCH_INDEX	Enables a property or dimension for record search.

### Example

This example enables a property (P\_Description) and a dimension (Winery) for record search, including wildcard search.

```
<RECSEARCH_INDEXES DICTIONARY_MAX_NGRAM_LENGTH="5 "
  DICTIONARY_WILDCARD="TRUE" INMEM_INDEX_THRESHOLD="512 "
  MAX_NGRAM_LENGTH="5" MAX_WORD_LENGTH="16 "
  MIN_WORD_LENGTH="3" MIN_WORD_OCCURRENCES="4">
  <RECSEARCH_INDEX NAME="P_Description">
    <SEARCH_INDEX>
      <WILDCARD_INDEX/>
      <POSITIONAL_INDEX/>
    </SEARCH_INDEX>
  </RECSEARCH_INDEX>
  <RECSEARCH_INDEX NAME="Winery" RECSEARCH_HIERARCHY="TRUE">
    <SEARCH_INDEX>
      <WILDCARD_INDEX/>
      <POSITIONAL_INDEX/>
    </SEARCH_INDEX>
  </RECSEARCH_INDEX>
```

## Refinement\_config.xml elements

The Refinement\_config.xml file configures the ranking, sorting, clustering, and collapsing behaviors of refinement values returned from the MDEX Engine.

Each refinement is specified in a REFINEMENTS element and those are grouped in the root element REFINEMENTS\_CONFIG.

### CLUSTERS

A CLUSTERS element enables clustering for a given dimension and sets the parameters of each cluster.

For more details on clusters, see the *Endeca Relationship Discovery Guide*.

#### DTD

```
<!ELEMENT CLUSTERS EMPTY>
<!ATTLIST CLUSTERS
  NAME                CDATA      #REQUIRED
  COHERENCE           CDATA      #IMPLIED
  REC_SAMPLE_SIZE     CDATA      #IMPLIED
  MAX_CLUSTERS        CDATA      #IMPLIED
  MAX_CLUSTER_SIZE    CDATA      #IMPLIED
  MAX_CLUSTER_OVERLAP CDATA      #IMPLIED
  MAX_REFINEMENT_PRECISION CDATA    #IMPLIED
>
```

#### Attributes

The following sections describe the CLUSTERS element's attributes.

##### NAME

Specifies the name of the dimension for which to enable clustering. The dimension must be a related terms dimension.

##### COHERENCE

Specifies a number that governs the decision of whether a set of terms is coherent enough to form a cluster. The number must be between 0 and 10, where 10 leads to the smallest clusters and 0 leads to the largest. That is, low values are permissive (do not demand much coherence) and will result in fewer larger clusters, while high values are strict and will result in more smaller clusters. Defaults to 5.

##### REC\_SAMPLE\_SIZE

Specifies how many records from the current navigation state the MDEX Engine should consider when computing clusters. The range is from 50 to 2000, and the default is 500. Lower numbers will result in less memory consumption and faster turnaround from the MDEX Engine, but could produce less accurate results.

##### MAX\_CLUSTERS

Specifies the maximum number of clusters that can be returned by the MDEX Engine. The range is from 2 to 10, with a default value of 6.

##### MAX\_CLUSTER\_SIZE

Specifies the maximum number of terms that can be in a cluster. The range is from 2 to 10, with a default value of 8.

##### MAX\_CLUSTER\_OVERLAP

Specifies the amount of overlap allowed between clusters. The range is from 0 to 10, with a default value of 5. If two clusters overlap (that is, if the document sets that each cluster maps to overlap), then the smaller cluster can be removed, depending on how big the overlap is. The value of this attribute dictates the overlap above which the smaller cluster is removed. If the value is set at 1, all smaller overlapping clusters are removed, even if they overlap by only one document; if the value is set at 10, only fully overlapping (subsumed) clusters are removed. A value of 0 will disable the overlapping cluster removal functionality.

### MAX\_REFINEMENT\_PRECISION

Specifies the maximum precision for a refinement which is to be considered for clustering. The value is a floating number between 0 and 1, with a default of 0.4. Refinements with higher precision will be discarded. Setting this value lower prevents clustering refinements which essentially summarize a query.

### Sub-elements

The CLUSTERS element has no sub-elements.

### Example

This example enables clustering on the dimension named Related Terms.

```
<REFINEMENTS_CONFIG>
  <REFINEMENTS NAME="Related Terms" SORT_TYPE="ALPHA">
    <DYNAMIC_RANKING COUNT="10" MORE="FALSE"
      TOP_REFINEMENTS_SORT="DEFAULT" TYPE="FREQUENCY"/>
  </REFINEMENTS>
  <CLUSTERS NAME="Related Terms" COHERENCE="4"
    MAX_CLUSTERS="5" REC_SAMPLE_SIZE="500" MAX_CLUSTER_SIZE="10"
    MAX_CLUSTER_OVERLAP="7" MAX_REFINEMENT_PRECISION="0.25"/>
</REFINEMENTS_CONFIG>
```

## DYNAMIC\_RANKING

A DYNAMIC\_RANKING element controls how refinements (dimension values) are returned from the dimension that contains this element.

If this element is included, then refinement ranking is enabled for this dimension.

### DTD

```
<!ELEMENT DYNAMIC_RANKING EMPTY>
<!ATTLIST DYNAMIC_RANKING
  COUNT          CDATA          #IMPLIED
  MORE           ( TRUE | FALSE ) #REQUIRED
  TYPE           ( FREQUENCY | WEIGHTED_FREQUENCY ) "FREQUENCY"
  TOP_REFINEMENTS_SORT ( DYNAMIC_RANK | DEFAULT ) "DYNAMIC_RANK"
>
```

### Attributes

The following sections describe the DYNAMIC\_RANKING element's attributes.

#### COUNT

Specifies the number of refinements (dimension values) that should be displayed for this dimension. If the number of refinements that are returned exceeds the value specified in COUNT, then a More link displays if MORE="TRUE".

#### MORE

When set to TRUE, the MDEX Engine computes the More refinement for dimensions and returns all of those dimension values. For example, if COUNT="100" and there are 1000 refinements returned, then clicking the More link displays the remaining 900.

When set to FALSE, the MDEX Engine computes the top refinements based on rank and returns those. The default value is TRUE.

### TYPE

Specifies how the ranking for a particular refinement value is determined. There are two valid values:

- FREQUENCY determines the refinement ranking based on the number of records that the refinement value produces. The default value of TYPE is FREQUENCY.
- WEIGHTED\_FREQUENCY determines a dimension value's rank by multiplying the number of records that it produces by the dimension value's static rank, which is assigned in dval\_ranks.xml.



**Note:** In the MDEX 6.2.0 release, the WEIGHTED\_FREQUENCY option has been deprecated.

### TOP\_REFINEMENTS\_SORT

Specifies the order in which refinements are returned:

- DYNAMIC\_RANK returns the refinements in refinement-rank order (refinements with higher rank are returned before refinements with lower rank).
- DEFAULT returns the refinements in the order specified by the SORT\_ORDER attribute of the parent REFINEMENTS element.

The default value is DYNAMIC\_RANK.

### Sub-elements

The DYNAMIC\_RANKING element has no sub-elements.

### Example

This example sets dynamic ranking values for the Winery dimension.

```
<REFINEMENTS_CONFIG>
  <REFINEMENTS NAME="Winery" SORT_TYPE="ALPHA">
    <DYNAMIC_RANKING COUNT="20" MORE="FALSE"
      TOP_REFINEMENTS_SORT="DEFAULT" TYPE="FREQUENCY" />
  </REFINEMENTS>
</REFINEMENTS_CONFIG>
```

## REFINEMENTS

The REFINEMENTS element specifies how dimensions behave when returned from the MDEX Engine as refinements.

### DTD

```
<!ELEMENT REFINEMENTS
  (
    STATS?
    , DYNAMIC_RANKING?
  )
>
<!ATTLIST REFINEMENTS
  NAME                CDATA                #REQUIRED
```

```

    SORT_TYPE           ( ALPHA | NUMERIC )   " ALPHA "
    SORT_ORDER          ( ASC | DESC )        " ASC "
    DVAL_COLLAPSE_THRESHOLD CDATA            #IMPLIED
  >

```

## Attributes

The following sections describe the REFINEMENTS element's attributes.

### NAME

Specifies a unique name for the refinement dimension.

### SORT\_TYPE

Specifies the order in which refinements from this dimension should be sorted when they are returned from the MDEX Engine:

- ALPHA sorts refinements alphabetically.
- NUMERIC sorts refinements numerically.

The default is ALPHA.

### SORT\_ORDER

Specifies the order in which refinements are returned:

- ASC specifies ascending order.
- DESC specifies descending order.

### DVAL\_COLLAPSE\_THRESHOLD

Specifies the minimum number of refinements that must be children of a dimension value from this dimension before the refinements are collapsed into their parent.

A collapsible dimension value within this dimension will not be collapsed unless its number of refinements exceeds the DVAL\_COLLAPSE\_THRESHOLD value. For example, if a dimension named "Author" has a DVAL\_COLLAPSE\_THRESHOLD="20" and a dimension value within Author (such as "A") is marked COLLAPSIBLE="TRUE", then a search that returns 20 or more children is collapsed to display "A". The user can then expand "A" and its refinements. If the search returns 19 records or fewer, all the results are displayed (uncollapsed).

## Sub-elements

The following table provides a brief overview of the REFINEMENTS sub-elements.

Sub-element	Brief description
STATS	Instructs the MDEX Engine to return statistics about refinements (dimension values) as part of the search query results.
DYNAMIC_RANKING	Controls how refinements (dimension values) are returned from the dimension that contains this element.

## Example

This example shows the refinement settings for two dimensions.

```

<REFINEMENTS_CONFIG>
  <REFINEMENTS NAME="Winery" SORT_TYPE="ALPHA" >
    <DYNAMIC_RANKING COUNT="20" MORE="FALSE"
      TOP_REFINEMENTS_SORT="DEFAULT" TYPE="FREQUENCY" />

```

```

</REFINEMENTS>
<REFINEMENTS DVAL_COLLAPSE_THRESHOLD="10 "
  NAME="Price Range" SORT_TYPE="NUMERIC">
</REFINEMENTS_CONFIG>

```

## REFINEMENTS\_CONFIG

A REFINEMENTS\_CONFIG element configures the ranking, sorting, clustering, and collapsing behaviors of refinement values returned from the MDEX Engine.

### DTD

```

<!ELEMENT REFINEMENTS_CONFIG
  ( COMMENT?
    , (CLUSTERS | REFINEMENTS)*
    , STATS?
  )
>

```

### Attributes

The REFINEMENTS\_CONFIG element has no attributes.

### Sub-elements

The following table provides a brief overview of the REFINEMENTS\_CONFIG sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.
CLUSTERS	Enables and configures clustering for dimensions.
REFINEMENTS	Specifies how dimensions behave when returned from the MDEX Engine as refinements.
STATS	Instructs the MDEX Engine to return statistics about refinements (dimension values) as part of the search query results.

### Example

This example shows the refinement settings for two dimensions.

```

<REFINEMENTS_CONFIG>
  <REFINEMENTS NAME="Winery" SORT_TYPE="ALPHA">
    <DYNAMIC_RANKING COUNT="20" MORE="FALSE"
      TOP_REFINEMENTS_SORT="DEFAULT" TYPE="FREQUENCY" />
  </REFINEMENTS>
  <REFINEMENTS DVAL_COLLAPSE_THRESHOLD="10 "
    NAME="Price Range" SORT_TYPE="NUMERIC">
</REFINEMENTS_CONFIG>

```

## STATS

A STATS element instructs the MDEX Engine to return statistics about refinements (dimension values) as part of the search query results.

### DTD

```
<!ELEMENT STATS EMPTY>
<!ATTLIST STATS
  NUM_RECORDS (TRUE | FALSE) "FALSE"
  NUM_REFINEMENTS_IN_GROUP (TRUE | FALSE) "FALSE"
  MAX_RECORDS_COUNT CDATA #IMPLIED
  RECORD_COUNT_DISABLE_THRESHOLD CDATA #IMPLIED
  DVAL_STATIC_RECORD_COUNT (TRUE | FALSE) "FALSE"
  DVAL_STATIC_RANK (TRUE | FALSE) #IMPLIED
  REFINEMENT_RANK_SCORE (TRUE | FALSE) #IMPLIED
>
```

### Attributes

The following sections describe the STATS element's attributes.

#### NUM\_RECORDS

Specifies whether the MDEX Engine should compute the number of records associated with each refinement. This number is returned as a dimension value property in the MDEX Engine query results. The default value of this attribute is FALSE.

#### NUM\_REFINEMENTS\_IN\_GROUP

Specifies whether the MDEX Engine should return the number of refinements in an expanded refinement group. This number is returned as a dimension value property in the MDEX Engine query results. The default value of this attribute is FALSE.

#### MAX\_RECORDS\_COUNT

Specifies the maximum number of records that should be returned for the record statistics on a refinement. This option is useful, if for example, you wish to know whether there are at least 1000 records for a particular refinement option, but you do not care if there are more than 1000. If you have enabled record statistics in the MDEX Engine indexer configuration file, reducing this number will improve the performance of the MDEX Engine. Note that this threshold applies to all dimensions. By default, there is no threshold for the record statistics computation.

#### RECORD\_COUNT\_DISABLE\_THRESHOLD

Specifies the maximum number of records in a result set above which record statistics should not be computed for the refinements. By default, there is no threshold for disabling the record statistics computation.

#### DVAL\_STATIC\_RECORD\_COUNT

Specifies whether the number of records associated with each dimension value should be returned as a dimension value property when that dimension value is returned by the MDEX Engine (for example, as a refinement). The number of records associated with each dimension value in this case is statically determined during indexing, and is the total number of records in the data set that have the dimension value. The default value is FALSE.

#### DVAL\_STATIC\_RANK

Specifies whether every dimension value's static rank should be returned as a property on the dimension value. The default value is FALSE.



**Note:** This attribute is reinstated starting with the MDEX Engine release 6.1.4. This attribute has been deprecated in 6.0.0-6.1.3 releases of MDEX Engine. If this attribute was used in releases of the MDEX Engine 6.0.0 through 6.1.3, the MDEX Engine ignored it and issued a warning about its presence in the file.

### REFINEMENT\_RANK\_SCORE

Specifies whether the MDEX Engine should return the rank of each refinement that is from a ranked dimension. If set to TRUE, then each refinement's rank is returned as a dimension value property. This attribute affects pre-computation and the run-time operation of the MDEX Engine. The default value is FALSE.

#### Sub-elements

STATS contains no sub-elements.

#### Example

This example shows a simple usage of STATS.

```
<STATS NUM_RECORDS="TRUE" MAX_RECORDS_COUNT="500" />
```

## Relrank\_strategies.xml elements

The Relrank\_strategies.xml file contains the relevance ranking strategies for an application.

The strategies are grouped in the root element RELRANK\_STRATEGIES. Each strategy is expressed in a RELRANK\_STRATEGY element, which in turn is made of individual relevance ranking modules such as RELRANK\_EXACT, RELRANK\_FIELD, and so on.

For more information about relevance ranking, see "Using Relevance Ranking" in the *Endeca MDEX Engine Development Guide*.

### RELRANK\_APPROXPHRASE

The RELRANK\_APPROXPHRASE element implements the Approximate Phrase relevance ranking module.

This module is similar to RELRANK\_PHRASE, except that in the higher stratum, only the first instance of an exact match of the user's phrase is considered, which improves system performance.



**Note:** The RELRANK\_APPROXPHRASE element is no longer supported. Use the RELRANK\_PHRASE element with the APPROXIMATE attribute instead.

#### DTD

```
<!ELEMENT RELRANK_APPROXPHRASE EMPTY>
```

#### Attributes

The RELRANK\_APPROXPHRASE element has no attributes.

#### Sub-elements

The RELRANK\_APPROXPHRASE element has no sub-elements.

## RELANK\_EXACT

The RELANK\_EXACT element implements the Exact relevance ranking module.

This module groups results into strata based on how well they match a query string, with the highest stratum containing results that match the user's query exactly. See "Using Relevance Ranking" in the *Endeca Development Guide* for details.

### DTD

```
<!ELEMENT RELANK_EXACT EMPTY>
```

### Attributes

The RELANK\_EXACT element has no attributes.

### Sub-elements

The RELANK\_EXACT element has no sub-elements.

### Example

In this example, the ranking strategy MyStrategy includes the RELANK\_EXACT element.

```
<RELANK_STRATEGY NAME="MyStrategy">
  <RELANK_STATIC NAME="Availability" ORDER="DESCENDING"/>
  <RELANK_EXACT/>
  <RELANK_STATIC NAME="Price" ORDER="ASCENDING"/>
</RELANK_STRATEGY>
```

## RELANK\_FIELD

The RELANK\_FIELD element implements the Field relevance ranking module.

This module assigns a score to each result based on the static rank of the dimension or property member of the search interface that caused the document to match the query. See "Using Relevance Ranking" in the *Endeca Development Guide* for details.

### DTD

```
<!ELEMENT RELANK_FIELD EMPTY>
```

### Attributes

The RELANK\_FIELD element has no attributes.

### Sub-elements

The RELANK\_FIELD element has no sub-elements.

### Example

In this example, the field module is included in a strategy called All\_Fields.

```
<RELANK_STRATEGY NAME="All_Fields">
  <RELANK_EXACT/>
  <RELANK_INTERP/>
</RELANK_STRATEGY>
```

```
<REL_RANK_FIELD />
</REL_RANK_STRATEGY >
```

## REL\_RANK\_FIRST

The REL\_RANK\_FIRST element implements the First relevance ranking module.

This module ranks documents by how close the query terms are to the beginning of the document. This module takes advantage of the fact that the closer something is to the beginning of a document, the more likely it is to be relevant. See "Using Relevance Ranking" in the *Endeca MDEX Engine Development Guide* for details.

### DTD

```
<!ELEMENT REL_RANK_FIRST EMPTY >
```

### Attributes

The REL\_RANK\_FIRST element has no attributes.

### Sub-elements

The REL\_RANK\_FIRST element has no sub-elements.

### Example

In this example, the ranking strategy All includes the First relevance ranking module.

```
<REL_RANK_STRATEGY NAME="All" >
  <REL_RANK_FIRST />
  <REL_RANK_INTERP />
  <REL_RANK_FIELD />
</REL_RANK_STRATEGY >
```

## REL\_RANK\_FREQ

The REL\_RANK\_FREQ element implements the Frequency relevance ranking module.

This module provides result scoring based on the frequency (number of occurrences) of the user's query terms in the result text. See "Using Relevance Ranking" in the *Endeca MDEX Engine Development Guide* for details.

### DTD

```
<!ELEMENT REL_RANK_FREQ EMPTY >
```

### Attributes

The REL\_RANK\_FREQ element has no attributes.

### Sub-elements

The REL\_RANK\_FREQ element has no sub-elements.

**Example**

This example implements a strategy called Frequency.

```
<RELRANK_STRATEGY NAME="Frequency">
  <RELRANK_FREQ/>
</RELRANK_STRATEGY>
```

**RELRANK\_GLOM**

The RELRANK\_GLOM element implements the Glom relevance ranking module.

This module ranks single-field matches ahead of cross-field matches. See "Using Relevance Ranking" in the *Endeca MDEX Engine Development Guide* for details.

**DTD**

```
<!ELEMENT RELRANK_GLOM EMPTY>
```

**Attributes**

The RELRANK\_GLOM element has no attributes.

**Sub-elements**

The RELRANK\_GLOM element has no sub-elements.

**Example**

This example implements a strategy called Single\_Field.

```
<RELRANK_STRATEGY NAME="Single_Field">
  <RELRANK_GLOM/>
</RELRANK_STRATEGY>
```

**RELRANK\_INTERP**

The RELRANK\_INTERP element implements the Interpreted (Interp) relevance ranking module.

This module provides a general-purpose strategy that assigns a score to each result document based on the query processing techniques used to obtain the match. Matching techniques considered include partial matching, cross-attribute matching, spelling correction, thesaurus, and stemming matching. See "Using Relevance Ranking" in the *Endeca MDEX Engine Development Guide* for details.

**DTD**

```
<!ELEMENT RELRANK_INTERP EMPTY>
```

**Attributes**

The RELRANK\_INTERP element has no attributes.

**Sub-elements**

The RELRANK\_INTERP element has no sub-elements.

### Example

In this example, the Interpreted module is included in a strategy called All\_Fields.

```
<REL_RANK_STRATEGY NAME="All_Fields">
  <REL_RANK_EXACT/>
  <REL_RANK_INTERP/>
  <REL_RANK_FIELD/>
</REL_RANK_STRATEGY>
```

## REL\_RANK\_MAXFIELD

The REL\_RANK\_MAXFIELD element implements the Maximum Field (Maxfield) relevance ranking module.

This module is similar to the Field strategy module, except it selects the static field-specific score of the highest-ranked field that contributed to the match. See "Using Relevance Ranking" in the *Endeca MDEX Engine Development Guide* for details.

### DTD

```
<!ELEMENT REL_RANK_MAXFIELD EMPTY>
```

### Attributes

The REL\_RANK\_MAXFIELD element has no attributes.

### Sub-elements

The REL\_RANK\_MAXFIELD element has no sub-elements.

### Example

This example implements a strategy called High\_Rank.

```
<REL_RANK_STRATEGY NAME="High_Rank">
  <REL_RANK_MAXFIELD/>
</REL_RANK_STRATEGY>
```

## REL\_RANK\_MODULE

The REL\_RANK\_MODULE element is used to refer to and compose other relevance ranking modules into strategies.

### DTD

```
<!ELEMENT REL_RANK_MODULE (REL_RANK_MODULE_PARAM*)>
<!ATTLIST REL_RANK_MODULE
  NAME          CDATA          #REQUIRED
>
```

### Attributes

The following section describes the REL\_RANK\_MODULE attribute.

#### NAME

NAME refers to another defined relevance ranking module.

### Sub-elements

The RELRANK\_MODULE element has no supported sub-elements. RELRANK\_MODULE\_PARAM is not supported.

### Example

In this example, a strategy called Best Price is defined. Later, this strategy is included in another strategy definition using the RELRANK\_MODULE element.

```

<RELKANK_STRATEGY NAME="Best Price">
  <RELKANK_STATIC NAME="Price" ORDER="ASCENDING"/>
</RELKANK_STRATEGY>
<RELKANK_STRATEGY NAME="MyStrategy">
  <RELKANK_STATIC NAME="Availability" ORDER="DESCENDING"/>
  <RELKANK_EXACT/>
  <RELKANK_MODULE NAME="Best Price"/>
</RELKANK_STRATEGY>

```

## RELRANK\_NTERMS

The RELRANK\_NTERMS element implements the Number of Terms (Nterms) relevance ranking module.

This module assigns a score to each result record based on the number of query terms that the result record matches. For example, in a three-word query, results that match all three words are ranked above results that match only two words, which are ranked above results that match only one word. See "Using Relevance Ranking" in the *Endeca MDEX Engine Development Guide* for details.

This module applies only to search modes where the number of results can vary in how many query terms they match. These search modes include matchpartial, matchany, matchallpartial, and matchallany. See "Using Search Modes" in the *Endeca MDEX Engine Basic Developer's Guide* for details.

### DTD

```

<!ELEMENT RELRANK_NTERMS EMPTY>

```

### Attributes

The RELRANK\_NTERMS element has no attributes.

### Sub-elements

The RELRANK\_NTERMS element has no sub-elements.

### Example

In this example, the Nterms module is included in a strategy called NumberOfTerms.

```

<RELKANK_STRATEGY NAME="NumberOfTerms">
  <RELKANK_NTERMS/>
</RELKANK_STRATEGY>

```

## RELRANK\_NUMFIELDS

The RELRANK\_NUMFIELDS element implements the Number of Fields (Numfields) relevance ranking module.

This module ranks results based on the number of fields in the associated search interface in which a match occurs. See "Using Relevance Ranking" in the *Endeca MDEX Engine Development Guide* for details.

### DTD

```
<!ELEMENT RELRANK_NUMFIELDS EMPTY>
```

### Attributes

The RELRANK\_NUMFIELDS element has no attributes.

### Sub-elements

The RELRANK\_NUMFIELDS element has no sub-elements.

### Example

This example implements the Numfields relevance ranking module.

```
<RELRANK_STRATEGY NAME="NumFields">
  <RELRANK_NUMFIELDS/>
</RELRANK_STRATEGY>
```

## RELRANK\_PHRASE

The RELRANK\_PHRASE element implements the Phrase relevance ranking module.

This module states that results containing the user's query as an exact phrase, or a subset of the exact phrase, should be considered more relevant than matches simply containing the user's search terms scattered throughout the text. Note that records that have the phrase are ranked higher than records which do not contain the phrase. See "Using Relevance Ranking" in the *Endeca MDEX Engine Development Guide* for details.

### DTD

```
<!ELEMENT RELRANK_PHRASE EMPTY>
<!ATTLIST RELRANK_PHRASE
  SUBPHRASE      ( TRUE | FALSE ) "FALSE"
  APPROXIMATE    ( TRUE | FALSE ) "FALSE"
  QUERY_EXPANSION ( TRUE | FALSE ) "FALSE"
>
```

### Attributes

The following sections describe the RELRANK\_PHRASE element's attributes.

#### SUBPHRASE

If set to TRUE, enables subphrasing, which ranks results based on the length of their subphrase matches.

If set to FALSE (the default), subphrasing is not enabled, which means that results are ranked into two strata: those that matched the entire phrase and those that did not.

#### APPROXIMATE

If set to TRUE, approximate matching is enabled. In this case, the Phrase module looks at a limited number of positions in each result that a phrase match could possibly exist, rather than all the positions. Only this limited number of possible occurrences is considered, regardless of whether there are later occurrences that are better, more relevant matches.

#### QUERY\_EXPANSION

If set to TRUE, enables query expansion, in which spelling correction, thesaurus, and stemming adjustments are applied to the original phrase. With query expansion enabled, the Phrase module ranks results that match a phrase's expanded forms in the same stratum as results that match the original phrase.

### Sub-elements

The RELRANK\_PHRASE element has no sub-elements.

### Example

This example of the Phrase module enables approximate matching and query expansion, and disables subphrasing.

```
<RELKANK_STRATEGY NAME="PhraseMatch">
  <RELKANK_PHRASE APPROXIMATE="TRUE "
    QUERY_EXPANSION="TRUE " SUBPHRASE="FALSE " />
</RELKANK_STRATEGY>
```

## RELKANK\_PROXIMITY

The RELKANK\_PROXIMITY element implements the Proximity relevance ranking module.

This module ranks how close the query terms are to each other in a document by counting the number of intervening words. See "Using Relevance Ranking" in the *Endeca MDEX Engine Development Guide* for details.

### DTD

```
<!ELEMENT RELKANK_PROXIMITY EMPTY>
```

### Attributes

The RELKANK\_PROXIMITY element has no attributes.

### Sub-elements

The RELKANK\_PROXIMITY element has no sub-elements.

### Example

This example implements a strategy called All that includes the Proximity module.

```
<RELKANK_STRATEGY NAME="All">
  <RELKANK_PROXIMITY />
  <RELKANK_INTERP />
  <RELKANK_FIELD />
</RELKANK_STRATEGY>
```

## RELKANK\_SPELL

The RELKANK\_SPELL element implements the Spell relevance ranking module.

This module ranks matches that do not require spelling correction ahead of spelling-corrected matches. See "Using Relevance Ranking" in the *Endeca MDEX Engine Development Guide* for details.

**DTD**

```
<!ELEMENT RELRANK_SPELL EMPTY>
```

**Attributes**

The RELRANK\_SPELL element has no attributes.

**Sub-elements**

The RELRANK\_SPELL element has no sub-elements.

**Example**

This example implements a strategy called TrueMatch.

```
<RELRANK_STRATEGY NAME="TrueMatch">
  <RELRANK_SPELL/>
</RELRANK_STRATEGY>
```

**RELRANK\_STATIC**

The RELRANK\_STATIC element implements the Static relevance ranking module.

This module assigns a constant score to each result, depending on the type of search operation performed. See "Using Relevance Ranking" in the *Endeca MDEX Engine Development Guide* for details.

**DTD**

```
<!ELEMENT RELRANK_FREQ EMPTY>
<!ATTLIST RELRANK_STATIC
  NAME CDATA #REQUIRED
  ORDER (ASCENDING|DESCENDING) #REQUIRED
>
```

**Attributes**

The following sections describe the RELRANK\_STATIC attributes.

**NAME**

Specifies the name of a property or dimension that used for static relevance ranking.

**ORDER**

Specifies how records should be sorted with respect to the specified property or dimension.

**Sub-elements**

The RELRANK\_STATIC element has no sub-elements.

**Example**

In this example, the BestPrice strategy consists of the Price dimension sorted from lowest to highest.

```
<RELRANK_STRATEGY NAME="BestPrice">
  <RELRANK_STATIC NAME="Price" ORDER="ASCENDING"/>
</RELRANK_STRATEGY>
```

## RELANK\_STRATEGIES

A RELANK\_STRATEGIES element contains any number of relevance ranking strategies for an application. Each strategy is specified in a RELRANK\_STRATEGY element.

### DTD

```
<!ELEMENT RELRANK_STRATEGIES
  ( COMMENT?
    ; RELRANK_STRATEGY*
  )
>
```

### Attributes

The RELRANK\_STRATEGIES element has no attributes.

### Sub-elements

The following table provides a brief overview of the RELRANK\_STRATEGIES sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <code>&lt;!-- ... --&gt;</code> .
RELRANK_STRATEGY	Contains a list of relevance ranking strategies that affect the order in which search results are returned to a user.

### Example

This example shows several strategies grouped under the root element RELRANK\_STRATEGIES.

```
<RELANK_STRATEGIES>
  <RELRANK_STRATEGY NAME="Bestseller Strategy">
    <RELRANK_STATIC NAME="Bestseller" ORDER="DESCENDING" />
  </RELRANK_STRATEGY>
  <RELRANK_STRATEGY NAME="Electronics Strategy">
    <RELRANK_FIELD />
    <RELRANK_EXACT />
    <RELRANK_INTERP />
    <RELRANK_STATIC NAME="Bestseller" ORDER="DESCENDING" />
    <RELRANK_STATIC NAME="Product_Name" ORDER="ASCENDING" />
  </RELRANK_STRATEGY>
</RELANK_STRATEGIES>
```

## RELRANK\_STRATEGY

The RELRANK\_STRATEGY element contains a list of relevance ranking strategies that affect the order in which search results are returned to a user.

Each sub-element of RELRANK\_STRATEGY represents a specific type of strategy. If you want several relevance ranking strategies to affect search result, then the order of the sub-elements, which represent the strategies, is significant. The order of the sub-elements defines the order in which the strategies are applied to the search results. See "Using Relevance Ranking" in the *Endeca MDEX Engine Development Guide* for details.

**DTD**

```

<!ELEMENT RELRANK_STRATEGY (
  RELRANK_STATIC
  | RELRANK_EXACT
  | RELRANK_PHRASE
  | RELRANK_APPROXPHRASE
  | RELRANK_GLOM
  | RELRANK_SPELL
  | RELRANK_FIELD
  | RELRANK_MAXFIELD
  | RELRANK_INTERP
  | RELRANK_FREQ
  | RELRANK_WFREQ
  | RELRANK_NTERMS
  | RELRANK_PROXIMITY
  | RELRANK_FIRST
  | RELRANK_NUMFIELDS
  | RELRANK_MODULE
) +>
<!ATTLIST RELRANK_STRATEGY
  NAME      CDATA      #REQUIRED
>

```

**Attributes**

The following section describes the RELRANK\_STRATEGY attribute.

**NAME**

Specifies the name of the strategy.

**Sub-elements**

The following table provides a brief overview of the RELRANK\_STRATEGY sub-elements.

Sub-element	Brief description
RELRANK_STATIC	Assigns a constant score to each result, depending on the type of search operation perform.
RELRANK_EXACT	Groups results into strata based on how well they match the query string, with the highest stratum containing results that match the user's query exactly.
RELRANK_PHRASE	Considers results containing the user's query as an exact phrase, or a subset of the exact phrase, to be more relevant than matches simply containing the user's search terms scattered throughout the text.
RELRANK_APPROXPHRASE	Not supported.
RELRANK_GLOM	Ranks single-field matches ahead of cross-field matches.
RELRANK_SPELL	Ranks true matches ahead of spelling-corrected matches.
RELRANK_FIELD	Assigns a score to each result based on the static rank of the dimension or property member of the search interface that caused the document to match the query.
RELRANK_MAXFIELD	Similar to the Field strategy, except it selects the static field-specific score of the highest-ranked field that contributed to the match.

Sub-element	Brief description
RELRANK_INTERP	A general-purpose strategy that assigns a score to each result document based on the query processing techniques used to obtain the match. Matching techniques considered include partial matching, cross-attribute matching, spelling correction, thesaurus, and stemming matching.
RELRANK_FREQ	Provides result scoring based on the frequency (number of occurrences) of the user's query terms in the result text.
RELRANK_WFREQ	Scores results based on the frequency of user query terms in the result, while weighing the individual query term frequencies for each result by the information content (overall frequency in the complete data set) of each query term.
RELRANK_NTERMS	Assigns a score to each result record based on the number of query terms that the result record matches.
RELRANK_PROXIMITY	Ranks how close the query terms are to each other in a document by counting the number of intervening words.
RELRANK_FIRST	Ranks documents by how close the query terms are to the beginning of the document.
RELRANK_NUMFIELDS	Ranks results based on the number of fields in the associated search interface in which a match occurs.
RELRANK_MODULE	Used to refer to other RELRANK elements and compose them into cohesive strategies.

### Example

This example presents a ranking strategy called Product\_Search\_Rank, which itself is composed of multiple strategies.

```
<RELRANK_STRATEGY NAME="Product_Search_Rank">
  <RELRANK_MODULE NAME="IsAvailable"/>
  <RELRANK_FIELD/>
  <RELRANK_PHRASE/>
  <RELRANK_MODULE NAME="BestPrice"/>
</RELRANK_STRATEGY>
```

## RELRANK\_WFREQ

The RELRANK\_WFREQ element implements the Weighted Frequency (Wfreq) relevance ranking module.

This module scores results based on the frequency of user query terms in the result, while weighing the individual query term frequencies for each result by the information content (overall frequency in the complete data set) of each query term. See "Using Relevance Ranking" in the *Endeca MDEX Engine Development Guide* for details.

### DTD

```
<!ELEMENT RELRANK_WFREQ EMPTY>
```

### Attributes

The RELRANK\_WFREQ element has no attributes.

**Sub-elements**

The RELRANK\_WFREQ element has no sub-elements.

**Example**

This example implements a strategy called Term\_Freq.

```
<RELRANK_STRATEGY NAME="Term_Freq">
  <RELRANK_WFREQ/>
</RELRANK_STRATEGY>
```

## Render\_config.xml elements

The Render\_config.xml file specifies the properties and dimensions that may be rendered on a record page or a record list page.

Each property or dimension is specified in a RENDER element. RENDER elements are grouped in the RENDER\_CONFIG parent element.

## RENDER

A RENDER element controls whether specified properties or dimensions may be rendered on a record list page or a record page.

**DTD**

```
<!ELEMENT RENDER EMPTY>
<!ATTLIST RENDER
  NAME                CDATA                #REQUIRED
  RENDER_PROD_LIST   ( FALSE | TRUE )   "TRUE "
  RENDER_PROD_PAGE   ( FALSE | TRUE )   "TRUE "
```

**Attributes**

The following sections describe the RENDER element's attributes.

**NAME**

Specifies the name of the property or dimension.

**RENDER\_PROD\_LIST**

Specifies whether the MDEX Engine returns this property or dimension along with a record on the record list page. The default value is TRUE.

**RENDER\_PROD\_PAGE**

Specifies whether the MDEX Engine returns this property or dimension along with a record on the product page. The default value is TRUE.

**Sub-elements**

The RENDER element has no sub-elements.

## Example

This example shows rendering configuration for a small number of properties.

```

<RENDER_CONFIG>
  <RENDER NAME="Camcorder_Features_Attr"
    RENDER_PROD_PAGE="TRUE" RENDER_PROD_LIST="TRUE" />
  <RENDER NAME="Camcorder_Formats_Attr"
    RENDER_PROD_PAGE="TRUE" RENDER_PROD_LIST="TRUE" />
  <RENDER NAME="Camcorder_Size_Attr"
    RENDER_PROD_PAGE="TRUE" RENDER_PROD_LIST="TRUE" />
  <RENDER NAME="Camera_Types_Attr"
    RENDER_PROD_PAGE="TRUE" RENDER_PROD_LIST="TRUE" />
</RECORD_SORT_CONFIG>

```

## RENDER\_CONFIG

A RENDER\_CONFIG element specifies the properties and dimensions that may be rendered on a product page or a record list page.

Each property or dimension is specified in a RENDER element.

### DTD

```

<!ELEMENT RENDER_CONFIG
  ( COMMENT?
    , RENDER*
  )
>

```

### Attributes

The RENDER\_CONFIG element has no attributes.

### Sub-elements

The following table provides a brief overview of the RENDER\_CONFIG sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.
RENDER	Controls whether specified properties or dimensions may be rendered on a record list page or a product page.

## Example

This example shows rendering configuration for a small number of properties.

```

<RENDER_CONFIG>
  <RENDER NAME="Camcorder_Features_Attr"
    RENDER_PROD_PAGE="TRUE" RENDER_PROD_LIST="TRUE" />
  <RENDER NAME="Camcorder_Formats_Attr"
    RENDER_PROD_PAGE="TRUE" RENDER_PROD_LIST="TRUE" />
  <RENDER NAME="Camcorder_Size_Attr"
    RENDER_PROD_PAGE="TRUE" RENDER_PROD_LIST="TRUE" />
  <RENDER NAME="Camera_Types_Attr"
    RENDER_PROD_PAGE="TRUE" RENDER_PROD_LIST="TRUE" />

```

```

    RENDER_PROD_PAGE="TRUE" RENDER_PROD_LIST="TRUE" />
</RENDER_CONFIG>

```

## Rollups.xml elements

The Rollups.xml file enables record aggregation based on a property or dimension value key.

By rolling up records based on a key, the MDEX Engine can handle multiple records as a single record. The ROLLUP element indicates a rollup key for record aggregation. ROLLUP elements are grouped in the ROLLUPS parent element. For more information, see "Creating Aggregated Records" in the *Endeca MDEX Engine Basic Development Guide*.

### ROLLUP

A ROLLUP element specifies a rollup key for record aggregation.

A rollup key may be either a dimension name or property name.

#### DTD

```

<!ELEMENT ROLLUP EMPTY>
<!ATTLIST ROLLUP
    NAME      CDATA      #REQUIRED
>

```

#### Attributes

The following section describes the ROLLUP element's attribute.

##### NAME

Specifies a dimension name or property name that is used as a rollup key.

#### Sub-elements

The ROLLUP element has no sub-elements.

#### Example

This example shows two rollup keys.

```

<ROLLUPS>
  <ROLLUP NAME="Manufacturer" />
  <ROLLUP NAME="Companies" />
</ROLLUPS>

```

### ROLLUPS

A ROLLUPS element contains any number of rollup keys indicated by ROLLUP elements.

#### DTD

```

<!ELEMENT ROLLUPS
    ( COMMENT?
    , ROLLUP*

```

```
)
>
```

### Attributes

The ROLLUPS element has no attributes.

### Sub-elements

The following table provides a brief overview of the ROLLUPS sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <code>&lt;!-- ... --&gt;</code> .
ROLLUP	Specifies a rollup key for record aggregation.

### Example

This example shows two rollup keys.

```
<ROLLUPS>
  <ROLLUP NAME="Manufacturer" />
  <ROLLUP NAME="Companies" />
</ROLLUPS>
```

## Search\_chars.xml elements

The Search\_chars.xml file contains searchable characters for your application.

The MDEX Engine treats such characters as searchable rather than treating them as punctuation. For more detailed information, see "Using Search Characters" in the *Endeca MDEX Engine Basic Development Guide*.

## SEARCH\_CHAR

The SEARCH\_CHAR element specifies a character that should be treated as a searchable character, instead of as punctuation.

### DTD

```
<!ELEMENT SEARCH_CHAR (#PCDATA)>
```

### Attributes

The SEARCH\_CHAR element has no attributes.

### Sub-elements

The SEARCH\_CHAR element has no sub-elements.

**Example**

In this example, to allow searching on "C++" and on "C#", you would specify:

```
<SEARCH_CHARS>
  <SEARCH_CHAR>+</SEARCH_CHAR>
  <SEARCH_CHAR>#</SEARCH_CHAR>
</SEARCH_CHARS>
```

**SEARCH\_CHARS**

A SEARCH\_CHARS element specifies the search characters enabled in your application.

Each search character is represented by a SEARCH\_CHAR element.

**DTD**

```
<!ELEMENT SEARCH_CHARS
  ( COMMENT?
    , SEARCH_CHAR*
  )
>
```

**Attributes**

The SEARCH\_CHARS element has no attributes.

**Sub-elements**

The following table provides a brief overview of the SEARCH\_CHARS sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.
SEARCH_CHAR	Specifies a character that should be treated as a searchable character, instead of as punctuation.

**Example**

In this example, to allow searching on "C++" and on "C#", you would specify:

```
<SEARCH_CHARS>
  <SEARCH_CHAR>+</SEARCH_CHAR>
  <SEARCH_CHAR>#</SEARCH_CHAR>
</SEARCH_CHARS>
```

**Searchindex.xml elements**

The Searchindex.xml file contains elements used to build indexes that serve to boost performance in searching.

The root element is SEARCH\_INDEX. Its sub-elements optimize indexes for particular types of searches and consequently, SEARCH\_INDEX may be found in other .xml files. For example, SEARCH\_INDEX may be found in dimsearch\_index.xml and recsearch\_indexes.xml.

## POSITIONAL\_INDEX

The POSITIONAL\_INDEX element specifies that a word position index should be created for the text collection containing this element.

Word position indices speed up the performance of phrase queries on large or off-line data sources such as property values with long descriptions or large amounts of document text.

### DTD

```
<!ELEMENT POSITIONAL_INDEX EMPTY>
```

### Attributes

The POSITIONAL\_INDEX element has no attributes.

### Sub-elements

The POSITIONAL\_INDEX element has no sub-elements.

### Example

This example enables positional indexing and wildcard indexing for searching on the Description property.

```
<RECSEARCH_INDEX NAME="Description">
  <SEARCH_INDEX>
    <WILDCARD_INDEX/>
    <POSITIONAL_INDEX/>
  </SEARCH_INDEX>
</RECSEARCH_INDEX>
```

## SEARCH\_INDEX

The SEARCH\_INDEX element specifies whether the wildcard index should be created for both record search and dimension search.

### DTD

```
<!ELEMENT SEARCH_INDEX
  ( WILDCARD_INDEX?
  , POSITIONAL_INDEX?
  , WORDAFFINITY_INDEX?
  )
>
<!ATTLIST SEARCH_INDEX
  INDEX_BUCKET_SIZE CDATA #IMPLIED
  INMEM_INDEX_THRESHOLD CDATA #IMPLIED
  TEXTSEARCH_SIZE_EST CDATA #IMPLIED
  MIN_WORD_OCCURRENCES CDATA #IMPLIED
  MIN_WORD_LENGTH CDATA #IMPLIED
  MAX_WORD_LENGTH CDATA #IMPLIED
>
```

### Attributes

The following sections describe the SEARCH\_INDEX element's attributes.

#### INDEX\_BUCKET\_SIZE

This attribute is deprecated.

### **INMEM\_INDEX\_THRESHOLD**

This attribute has been moved to RECSEARCH\_INDEXES and DIMSEARCH\_INDEX.

### **TEXTSEARCH\_SIZE\_EST**

This attribute is deprecated.

### **MIN\_WORD\_OCCURRENCES**

This attribute has been moved to RECSEARCH\_INDEXES and DIMSEARCH\_INDEX.

### **MIN\_WORD\_LENGTH**

This attribute has been moved to RECSEARCH\_INDEXES and DIMSEARCH\_INDEX.

### **MAX\_WORD\_LENGTH**

This attribute has been moved to RECSEARCH\_INDEXES and DIMSEARCH\_INDEX.

### **Sub-elements**

The following table provides a brief overview of the SEARCH\_INDEX sub-elements. Note that the WORDAFFINITY\_INDEX element is deprecated.

Sub-element	Brief description
WILDCARD_INDEX	Specifies that a wildcard index should be created for the text collection that contains this element.
POSITIONAL_INDEX	Specifies that a word position index should be created for the text collection that contains this element.

### **Example**

This example enables positional indexing and wildcard indexing for searching on the Description property.

```
<RECSEARCH_INDEX NAME="Description">
  <SEARCH_INDEX>
    <WILDCARD_INDEX/>
    <POSITIONAL_INDEX/>
  </SEARCH_INDEX>
</RECSEARCH_INDEX>
```

## **WILDCARD\_INDEX**

The WILDCARD\_INDEX element specifies that a wildcard index should be created for the text collection that contains it.

In addition, this element allows you to control the behavior of the construction of the wildcard index itself. Wildcard indices enable wildcard search, which is the ability to match user query terms to fragments of words in indexed text. The fragments themselves are referred to as *ngrams*.

### **DTD**

```
<!ELEMENT WILDCARD_INDEX EMPTY>
<!ATTLIST WILDCARD_INDEX
  MAX_NGRAM_LENGTH          CDATA          #IMPLIED
  DICTIONARY_WILDCARD      ( TRUE | FALSE ) #IMPLIED
```

```

    DICTIONARY_MAX_NGRAM_LENGTH CDATA #IMPLIED
  >

```

### Attributes

The following sections describe the WILDCARD\_INDEX element's attributes.

#### MAX\_NGRAM\_LENGTH

Specifies the maximum ngram length that should be indexed. All substrings of this length or shorter will be indexed. If a user does a wildcard search with a string that is less than or equal to this length, then exact results can be returned directly from the index. If the wildcard search includes a substring longer than this length, then the results returned from the index will be post-processed so that false positives are eliminated.

#### DICTIONARY\_WILDCARD

Specifies whether a wildcard index should be generated for the dictionary of words for the text collection containing this element. The default value of this attribute is TRUE when the containing element consists of off-line documents and FALSE otherwise.

#### DICTIONARY\_MAX\_NGRAM\_LENGTH

Specifies the maximum ngram length that should be indexed in the dictionary wildcard index.

### Sub-elements

The WILDCARD\_INDEX element has no sub-elements.

### Example

This example enables positional indexing and wildcard indexing for searching on the Description property.

```

<RECSEARCH_INDEX NAME="Description">
  <SEARCH_INDEX>
    <WILDCARD_INDEX/>
    <POSITIONAL_INDEX/>
  </SEARCH_INDEX>
</RECSEARCH_INDEX>

```

## Search\_interface.xml elements

The Search\_interface.xml contains elements used to build and configure search interfaces.

The file's root element is SEARCH\_INTERFACE. Search interfaces control record search behavior for groups of one or more properties or dimensions. For more information, see "About Search Interfaces" in the *Endeca Development Guide*.

### AUTO\_SUGGEST

The AUTO\_SUGGEST element specifies how to configure automatic spelling correction for either record searches or dimension searches.

If you want to use automatic spelling correction in record searches, you must first enable it by setting ENABLE\_AUTO\_SUGGEST to TRUE in the RECSEARCH\_CONFIG element. If you want to use automatic spelling correction in dimension searches, you must first enable it by setting ENABLE\_AUTO\_SUGGEST to TRUE in the DIMSEARCH\_CONFIG element.

For more information about spelling corrections, see "Using Spelling Correction and Did You Mean" in the *Endeca MDEX Engine Development Guide*.

### DTD

```
<!ELEMENT AUTO_SUGGEST (SEARCH_CORRECTION_PARAMETERS)>
```

### Attributes

The AUTO\_SUGGEST element has no attributes.

### Sub-elements

The following table provides a brief overview of the AUTO\_SUGGEST sub-element.

Sub-element	Brief description
SEARCH_CORRECTION_PARAMETERS	Specifies general spelling correction parameters for Did You Mean and automatic spelling correction.

## DID\_YOU\_MEAN

The DID\_YOU\_MEAN element specifies how to configure explicit alternatives for search terms as a part of spelling correction.

If you want to use Did You Mean alternative terms as a part of spelling correction, you must first enable Did You Mean by setting ENABLE\_DID\_YOU\_MEAN to TRUE in the RECSEARCH\_CONFIG element. For more information about spelling corrections, see "Using Spelling Correction and Did You Mean" in the *Endeca Development Guide*.

### DTD

```
<!ELEMENT DID_YOU_MEAN (SEARCH_CORRECTION_PARAMETERS)>
```

### Attributes

The DID\_YOU\_MEAN element has no attributes.

### Sub-elements

The following table provides a brief overview of the DID\_YOU\_MEAN sub-element.

Sub-element	Brief description
SEARCH_CORRECTION_PARAMETERS	Specifies general spelling correction parameters for Did You Mean and automatic spelling correction.

## MEMBER\_NAME

The MEMBER\_NAME element specifies the name of a property or dimension that is part of a SEARCH\_INTERFACE.

For information on search interfaces, see "About Search Interfaces" in the *Endeca MDEX Engine Basic Development Guide*.

**DTD**

```
<!ELEMENT MEMBER_NAME (#PCDATA)>
<!ATTLIST MEMBER_NAME
  RELEVANCE_RANK      CDATA      #IMPLIED
  SNIPPET_SIZE        CDATA      "0"
>
```

**Attributes**

The following sections describe the MEMBER\_NAME element's attributes.

**RELEVANCE\_RANK**

RELEVANCE\_RANK is an unsigned integer that specifies the relevance rank of a match on the specified dimension or property.

**SNIPPET\_SIZE**

The presence of SNIPPET\_SIZE enables snipping for a MEMBER\_NAME and the value of SNIPPET\_SIZE specifies maximum number of words a snippet can contain. Omitting this attribute or setting its value equal to zero disables snipping. For more information, see "Using Snipping in Record Searches" in the *Endeca Development Guide*.

**Sub-elements**

The MEMBER\_NAME element has no sub-elements.

**Example**

In the following example, four properties are listed in MEMBER\_NAME elements, each with its own relevance rank.

```
<SEARCH_INTERFACE CROSS_FIELD_BOUNDARY="NEVER"
  CROSS_FIELD_RELEVANCE_RANK="0"
  DEFAULT_RELRANK_STRATEGY="All" NAME="All">
  <MEMBER_NAME RELEVANCE_RANK="4">P_WineType</MEMBER_NAME>
  <MEMBER_NAME RELEVANCE_RANK="3">P_Name</MEMBER_NAME>
  <MEMBER_NAME RELEVANCE_RANK="2">P_Winery</MEMBER_NAME>
  <MEMBER_NAME RELEVANCE_RANK="1">P_Description</MEMBER_NAME>
</SEARCH_INTERFACE>
```

**PARTIAL\_MATCH**

The PARTIAL\_MATCH element specifies if partial query matches should be supported for the SEARCH\_INTERFACE that contains this element.

For details about searching and search modes, see "About Search Modes" in the *Endeca MDEX Engine Basic Development Guide*.

**DTD**

```
<!ELEMENT PARTIAL_MATCH EMPTY>
<!ATTLIST PARTIAL_MATCH
  MIN_WORDS_INCLUDED  CDATA      #IMPLIED
  MAX_WORDS_OMITTED   CDATA      #IMPLIED
>
```

## Attributes

The following sections describe the PARTIAL\_MATCH element's attributes.

### MIN\_WORDS\_INCLUDED

Specifies that search results match at least this number of terms in the search query. This value must be an integer greater than zero. The default value of this attribute is one.

### MAX\_WORDS\_OMITTED

Specifies the maximum number of query terms that may be ignored in the search query. This value must be a non-negative integer. If set to zero or left unspecified, any number of words may be omitted (i.e., there is no maximum). The default value of this attribute is two.

## Sub-elements

The PARTIAL\_MATCH element has no sub-elements.

## Example

In this example, the search interface is subject to partial matching in which at least two of the words in the search query are included, and no more than one is omitted.

```
<SEARCH_INTERFACE CROSS_FIELD_BOUNDARY="ALWAYS "
  CROSS_FIELD_RELEVANCE_RANK="0 "
  DEFAULT_RELRANK_STRATEGY="AllPhrase" NAME="AllPhrase">
  <MEMBER_NAME RELEVANCE_RANK="2">P_Body</MEMBER_NAME>
  <MEMBER_NAME RELEVANCE_RANK="1">P_Description</MEMBER_NAME>
  <PARTIAL_MATCH MAX_WORDS_OMITTED="1" MIN_WORDS_INCLUDED="2" />
</SEARCH_INTERFACE>
```

## SEARCH\_CORRECTION\_PARAMETERS

A SEARCH\_CORRECTION\_PARAMETERS element specifies general spelling correction parameters for Did You Mean and automatic spelling correction.

For more information about spelling corrections, see "Using Spelling Correction and Did You Mean" in the *Endeca Development Guide*.

## DTD

```
<!ELEMENT SEARCH_CORRECTION_PARAMETERS EMPTY>
<!ATTLIST SEARCH_CORRECTION_PARAMETERS
  HITS_THRESHOLD          CDATA          #IMPLIED
  HITS_GAIN_THRESHOLD    CDATA          #IMPLIED
  SPELLING_SCORE_THRESHOLD CDATA          #IMPLIED
  MAX_SUGGESTIONS_RETURNED CDATA          #IMPLIED
  MAX_WORDS_TO_REQUEST   CDATA          #IMPLIED
  MAX_WORDS_TO_USE       CDATA          #IMPLIED
  WORD_BREAK_ANALYSIS    ( TRUE | FALSE ) #IMPLIED
  CROSS_FIELD            ( TRUE | FALSE ) #IMPLIED
>
```

## Attributes

The following sections describe the SEARCH\_CORRECTION\_PARAMETERS element's attributes.

### HITS\_THRESHOLD

Specifies the maximum number of hits for which Did You Mean suggestions should be generated. When a very large number of hits are returned, computing the Did You Mean alternatives may be computationally expensive. The default value is 20.

#### **HITS\_GAIN\_THRESHOLD**

Specifies the minimum percentage gain in hits in order to consider an alternative valid. The default value is 10%. This value should be a decimal number between 0.0 and 1.0.

#### **SPELLING\_SCORE\_THRESHOLD**

Specifies the threshold correction score for words used by the Did You Mean engine. The default value is 150.

#### **MAX\_SUGGESTIONS\_RETURNED**

Specifies the maximum number of per-word spelling corrections to compute for each term in the search query. The default value is 3.

#### **MAX\_WORDS\_TO\_REQUEST**

Specifies the maximum number of per-word spelling corrections to request from the spelling engine for each term in the search query. The default value is 10.

#### **MAX\_WORDS\_TO\_USE**

Specifies the maximum number of highest-scored per-word spelling suggestions to use from among the available suggestions in the spelling engine. The default value is 10.

#### **WORD\_BREAK\_ANALYSIS**

Specifies whether Did You Mean considers alternate word separation points for the query in order to generate suggestions for spelling correction. For example, if a user searches for "Homemade cookies" then the engine may also consider separations like "Home made cookies." The default value is TRUE.

#### **CROSS\_FIELD**

Specifies that cross-property suggestions should be allowed and that they should be counted as matches when evaluating the frequencies of suggestions. Normally, suggestions must match results in a single property or dimension value.

#### **Sub-elements**

The SEARCH\_CORRECTION\_PARAMETERS element has no sub-elements.

## **SEARCH\_INTERFACE**

The SEARCH\_INTERFACE is a named collection of dimensions and/or properties.

Both dimensions and properties can co-exist in a SEARCH\_INTERFACE. The dimensions and/or properties in the group are specified in MEMBER\_NAME elements.

If a property or dimension is not included in any SEARCH\_INTERFACE element, then an implicit SEARCH\_INTERFACE element is created with the same name as the property or dimension and that single property or dimension as its only member. The value for the CROSS\_FIELD\_RELEVANCE\_RANK is set to 0.

#### **DTD**

```
<!ELEMENT SEARCH_INTERFACE
  ( MEMBER_NAME+
    , PARTIAL_MATCH?
    , AUTO_SUGGEST?
    , DID_YOU_MEAN?
```

```

)
>
<!ATTLIST SEARCH_INTERFACE
  NAME                CDATA          #REQUIRED
  DEFAULT_RELRANK_STRATEGY CDATA          #IMPLIED
  CROSS_FIELD_RELEVANCE_RANK CDATA          #IMPLIED
  CROSS_FIELD_BOUNDARY (ALWAYS
                       |ON_FAILURE
                       |NEVER)        "NEVER"
  STRICT_PHRASE_MATCH (TRUE|FALSE) #IMPLIED
>

```

## Attributes

The following sections describe the SEARCH\_INTERFACE element's attributes.

### NAME

A unique name for this search interface.

### DEFAULT\_RELRANK\_STRATEGY

For record search, a default relevance scoring function assigned to a SEARCH\_INTERFACE. If you are working in Endeca Developer Studio and have applied ranking modules to a search interface, this attribute is populated with a strategy based on the name of the search interface in question. For example, if your search interface is called Flavors, the DEFAULT\_RELRANK\_STRATEGY attribute has the value "Flavors\_strategy".

### CROSS\_FIELD\_RELEVANCE\_RANK

Specifies the relevance rank score for cross-field matches. The value should be an unsigned 32-bit integer. The default value for CROSS\_FIELD\_RELEVANCE\_RANK is 0.

### CROSS\_FIELD\_BOUNDARY

Specifies when the search engine should try to match search queries across dimension/property boundaries, but within the members of the SEARCH\_INTERFACE. If its value is set to ON\_FAILURE, then the search engine will only try to match queries across dimension/property boundaries if it fails to find any match within a single dimension/property. If its value is set to ALWAYS, then the engine will always look for matches across dimension/property boundaries, in addition to matches within a dimension/property.

By default, the MDEX Engine will not look across boundaries for matches.

### STRICT\_PHRASE\_MATCH

Specifies that the MDEX Engine should interpret a query strictly when comparing white space in the query with punctuation in the source text. If set to FALSE, partial word tokens connected in the source text by punctuation can be matched to a phrase query where the partial tokens are separated by spaces instead of matching punctuation. The default value of this attribute is TRUE.

## Sub-elements

The following table provides a brief overview of the SEARCH\_INTERFACE sub-elements.

Sub-element	Brief description
MEMBER_NAME	Specifies the name of a property or dimension that is part of a SEARCH_INTERFACE.
PARTIAL_MATCH	Specifies if partial query matches should be supported for the SEARCH_INTERFACE that contains this element.

Sub-element	Brief description
AUTO_SUGGEST	Specifies how to configure automatic spelling correction for either record searches or dimension searches.
DID_YOU_MEAN	Specifies how to configure explicit alternatives for search terms as a part of spelling correction.

### Example

This example establishes a search interface called AllFields, which contains four members.

```
<SEARCH_INTERFACE CROSS_FIELD_BOUNDARY="NEVER"
  CROSS_FIELD_RELEVANCE_RANK="0"
  DEFAULT_RELRANK_STRATEGY="All" NAME="AllFields">
  <MEMBER_NAME RELEVANCE_RANK="4">P_WineType</MEMBER_NAME>
  <MEMBER_NAME RELEVANCE_RANK="3">P_Name</MEMBER_NAME>
  <MEMBER_NAME RELEVANCE_RANK="2">P_Winery</MEMBER_NAME>
  <MEMBER_NAME RELEVANCE_RANK="1">P_Description</MEMBER_NAME>
</SEARCH_INTERFACE>
```

## Spell\_config.xml elements

The Spell\_config.xml file specifies which spelling correction dictionary, or combination of dictionaries, and which languages the MDEX Engines uses to provide spelling correct suggestions.

The root element of Spell\_config.xml is SPELL\_CONFIG. Through its sub-element SPELL\_ENGINE, you can configure the aspell or espell dictionary combinations and corresponding language codes. For more information, see "Using Spelling Correction and Did You Mean" in the *MDEX Engine Development Guide*.



**Note:** You cannot create this file in Developer Studio. Use a text editor to create it, then place the file in the directory where the project XML files reside.

## ASPELL

The ASPELL element instructs the MDEX Engine to correct queries using the Aspell dictionary.

Aspell is a phonetic spelling dictionary that corrects only English-like words, that includes letters A through Z but excludes accented letters and numbers. The English-like set includes a large subset of words in languages other than English, for example German and Spanish. If desired, you can set the LANGUAGES attribute to restrict corrections to English only.

### DTD

```
<!ELEMENT ASPELL EMPTY>
<!ATTLIST ASPELL
  LANGUAGES CDATA " "
>
```

### Attributes

The following section describes the ASPELL element's attribute.

#### LANGUAGES

Specifies a two-character language code that restricts spelling correction to a particular language. For the ASPELL element, the meaningful value of this attribute is `en` for English. If no value is provided, Aspell corrects using relevant English-like words.

### Sub-elements

The ASPELL element has no sub-elements.

### Example

This example uses the Aspell dictionary to correct English queries but not queries in other languages.

```
<SPELL_CONFIG>
  <SPELL_ENGINE>
    <ASPELL LANGUAGES="en" />
  </SPELL_ENGINE>
</SPELL_CONFIG>
```

## DICT\_PER\_LANGUAGE

The DICT\_PER\_LANGUAGE element creates a separate dictionary for each language.

Having separate dictionaries allows you to restrict spelling correction suggestions on a per-language basis. This is useful if, for example, you do not want English queries to produce spelling suggestions in Spanish.

### DTD

```
<!ELEMENT DICT_PER_LANGUAGE (ESPELL)>
<!ATTLIST DICT_PER_LANGUAGE
  LANGUAGES CDATA " "
>
```

### Attributes

The following section describes the DICT\_PER\_LANGUAGE element's attribute.

### LANGUAGES

Specifies a language code or a comma-separated list of language codes that restrict spelling correction to a particular language or a group of languages. ISO 639 lists the valid language codes.

Common examples include the following: `en` for English, `es` for Spanish, `fr` for French, `de` for German, `ja` for Japanese, and `ko` for Korean. Note that Chinese has two available codes: `zh-CN` for simplified Chinese and `zh-TW` for traditional Chinese. If no value is provided, Espell corrects using any relevant language.

### Sub-elements

The following table provides a brief overview of the DICT\_PER\_LANGUAGE sub-element.

Sub-element	Brief description
ESPELL	Instructs the MDEX Engine to correct queries using the Espell dictionary.

## Example

The following example creates separate spelling dictionaries for each language, using Aspell for English and Espell for all other languages.

```
<SPELL_CONFIG>
  <SPELL_ENGINE>
    <FIRST_DICT>
      <ASPELL LANGUAGES="en" />
      <DICT_PER_LANGUAGE>
        <ESPELL />
      </DICT_PER_LANGUAGE>
    </FIRST_DICT>
  </SPELL_ENGINE>
</SPELL_CONFIG>
```

## ESPELL

The ESPELL element instructs the MDEX Engine to correct queries using the Espell dictionary.

Espell is a non-phonetic spelling dictionary that corrects for words in any language including non-alphabetic words (i.e., words containing numbers or punctuation). If desired, you can set the LANGUAGES attribute to restrict corrections to a specific language or a set of languages.

### DTD

```
<!ELEMENT ESPELL EMPTY>
<!ATTLIST ESPELL
  LANGUAGES CDATA " "
>
```

### Attributes

The following section describes the ESPELL element's attribute.

#### LANGUAGES

Specifies a language code or a comma-separated list of language codes that restrict spelling correction to a particular language or a group of languages. ISO 639 lists the valid language codes.

Common examples include the following: `en` for English, `es` for Spanish, `fr` for French, `de` for German, `ja` for Japanese, and `ko` for Korean. Note that Chinese has a two available codes: `zh-CN` for simplified Chinese and `zh-TW` for traditional Chinese. If no value is provided, Espell corrects using any relevant language.

### Sub-elements

The ESPELL element has no sub-elements.

## Example

This example uses Espell to correct queries to English, Spanish, or French but no other languages.

```
<SPELL_CONFIG>
  <SPELL_ENGINE>
    <ESPELL LANGUAGES="en,es,fr" />
  </SPELL_ENGINE>
</SPELL_CONFIG>
```

## FIRST\_DICT

The FIRST\_DICT element combines two or more spelling dictionaries.

The FIRST\_DICT element also allows the first dictionary that can correct a word to provide spelling corrections. The remaining dictionaries do not process the word. For example, if you have Aspell as the first dictionary and Espell as the second, then words that Aspell cannot correct are handed to the Espell dictionary. In this configuration, the Espell engine would not process or suggest possible alternatives for any words corrected by Aspell. See the example below.

### DTD

```
<!ELEMENT FIRST_DICT
  ( ASPELL
  | ESPELL
  | UNION_DICTS
  | FIRST_DICT
  | DICT_PER_LANGUAGE
  )+
>
<!ATTLIST FIRST_DICT
  LANGUAGES CDATA " "
>
```

### Attributes

The following section describes the FIRST\_DICT element's attribute.

#### LANGUAGES

Specifies a language code or a comma-separated list of language codes that restrict spelling correction to a particular language or a group of languages. ISO 639 lists the valid language codes.

Common examples include the following: `en` for English, `es` for Spanish, `fr` for French, `de` for German, `ja` for Japanese, and `ko` for Korean. Note that Chinese has a two available codes: `zh-CN` for simplified Chinese and `zh-TW` for traditional Chinese. If no value is provided, Espell corrects using any relevant language.

### Sub-elements

The following table provides a brief overview of the FIRST\_DICT sub-elements.

Sub-element	Brief description
ASPELL	Instructs the MDEX Engine to correct queries using the Aspell dictionary.
ESPELL	Instructs the MDEX Engine to correct queries using the Espell dictionary.
UNION_DICTS	Combines two or more spelling dictionaries and allows each dictionary to process all words for spelling correction.
FIRST_DICT	Combines two or more spelling dictionaries and allows the first dictionary that can correct a word to provide spelling corrections.
DICT_PER_LANGUAGE	Creates a separate dictionary for each language. This allows you to restrict spelling correction suggestions on a per-language basis.

## Example

This example shows Aspell as the first dictionary and Espell as the second. Words that Aspell cannot correct are handed to the Espell dictionary. The Espell engine would not process or suggest possible alternatives for any words corrected by Aspell.

```
<SPELL_CONFIG>
  <SPELL_ENGINE>
    <FIRST_DICT>
      <ASPELL LANGUAGES="en" />
      <DICT_PER_LANGUAGE>
        <ESPELL />
      </DICT_PER_LANGUAGE>
    </FIRST_DICT>
  </SPELL_ENGINE>
</SPELL_CONFIG>
```

## SPELL\_CONFIG

A SPELL\_CONFIG element configures the global use of spelling correction dictionaries and languages for language-specific spelling correction.

### DTD

```
<!ELEMENT SPELL_CONFIG
  ( COMMENT?
    , SPELL_ENGINE?
  )
>
```

### Attributes

The SPELL\_CONFIG elements has no attributes.

### Sub-elements

The following table provides a brief overview of the SPELL\_CONFIG sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.
SPELL_ENGINE	Specifies whether to use Aspell, espell, or combinations of the two spelling dictionaries to make spelling corrections.

## Example

This example shows Aspell as the first dictionary and Espell as the second. Words that Aspell cannot correct are handed to the Espell dictionary. The Espell engine would not process or suggest possible alternatives for any words corrected by Aspell.

```
<SPELL_CONFIG>
  <SPELL_ENGINE>
    <FIRST_DICT>
      <ASPELL LANGUAGES="en" />
      <DICT_PER_LANGUAGE>
```

```

    <ESPELL/>
  </DICT_PER_LANGUAGE>
</FIRST_DICT>
</SPELL_ENGINE>
</SPELL_CONFIG>

```

## SPELL\_ENGINE

A SPELL\_ENGINE element specifies whether to use aspell, espell, or combinations of the two spelling dictionaries to make spelling corrections.

There are five possibilities to configure an application's spelling dictionaries:

- ASPELL on its own to correct only English-like words.
- ESPELL on its own to correct words in any language including non-alphabetic words.
- UNION\_DICTS combines the spelling suggestions from two or more occurrences of ASPELL and ESPELL.
- FIRST\_DICT returns suggestions from the first spelling dictionary that can correct a given word.
- DICT\_PER\_LANGUAGE uses a separate dictionary for each language.

All sub-elements of SPELL\_ENGINE have a LANGUAGES attribute that causes the spelling dictionary to correct to the language specified and to ignore words in other languages. For more information, see "Using Spelling Correction and Did You Mean" in the *MDEX Engine Development Guide*.

### DTD

```

<!ELEMENT SPELL_ENGINE
  (
    ASPELL
    |
    ESPELL
    |
    UNION_DICTS
    |
    FIRST_DICT
    |
    DICT_PER_LANGUAGE
  )
>

```

### Attributes

The SPELL\_ENGINE element has no attributes.

### Sub-elements

The following table provides a brief overview of the SPELL\_ENGINE sub-elements.

Sub-element	Brief description
ASPELL	Instructs the MDEX Engine to correct queries using the Aspell dictionary.
ESPELL	Instructs the MDEX Engine to correct queries using the Espell dictionary.
UNION_DICTS	Combines two or more spelling dictionaries and allows each dictionary to process all words for spelling correction.
FIRST_DICT	Combines two or more spelling dictionaries and allows the first dictionary that can correct a word to provide spelling corrections.
DICT_PER_LANGUAGE	Creates a separate dictionary for each language. This allows you to restrict spelling correction suggestions on a per-language basis.

## Example

This example shows Aspell as the first dictionary and Espell as the second. Words that Aspell cannot correct are handed to the Espell dictionary. The Espell engine would not process or suggest possible alternatives for any words corrected by Aspell.

```
<SPELL_CONFIG>
  <SPELL_ENGINE>
    <FIRST_DICT>
      <ASPELL LANGUAGES="en" />
      <DICT_PER_LANGUAGE>
        <ESPELL />
      </DICT_PER_LANGUAGE>
    </FIRST_DICT>
  </SPELL_ENGINE>
</SPELL_CONFIG>
```

## UNION\_DICTS

The UNION\_DICTS element combines two or more spelling dictionaries and allows each dictionary to process all words for spelling correction.

For example, if you union Aspell and Espell, then Aspell processes English-like words, while Espell processes all words. Because the two spell engines have different strengths, this is different behavior from simply letting Espell process all words. For more information, see "Using Spelling Correction and Did You Mean" in the *MDEX Engine Development Guide*.

## DTD

```
<!ELEMENT UNION_DICTS
  (
    ASPELL
  |
    ESPELL
  |
    UNION_DICTS
  |
    FIRST_DICT
  |
    DICT_PER_LANGUAGE
  )+
>
<!ATTLIST UNION_DICTS
  LANGUAGES CDATA " "
>
```

## Attributes

The following section describes the UNION\_DICTS element's attribute.

### LANGUAGES

Specifies a language code or a comma-separated list of language codes that restrict spelling correction to a particular language or a group of languages. ISO 639 lists the valid language codes.

Common examples include the following: `en` for English, `es` for Spanish, `fr` for French, `de` for German, `ja` for Japanese, and `ko` for Korean. Note that Chinese has a two available codes: `zh-CN` for simplified Chinese and `zh-TW` for traditional Chinese. If no value is provided, Espell corrects using any relevant language.

## Sub-elements

The following table provides a brief overview of the UNION\_DICTS sub-elements.

Sub-element	Brief description
ASPELL	Instructs the MDEX Engine to correct queries using the Aspell dictionary.
ESPELL	Instructs the MDEX Engine to correct queries using the Espell dictionary.
UNION_DICTS	Combines two or more spelling dictionaries and allows each dictionary to process all words for spelling correction.
FIRST_DICT	Combines two or more spelling dictionaries and allows the first dictionary that can correct a word to provide spelling corrections.
DICT_PER_LANGUAGE	Creates a separate dictionary for each language. This allows you to restrict spelling correction suggestions on a per-language basis.

### Example

The following example shows a union of Aspell and Espell, where Aspell processes English-like words, while Espell processes all words.

```
<SPELL_CONFIG>
  <SPELL_ENGINE>
    <UNION_DICTS>
      <ASPELL/>
      <ESPELL/>
    </UNION_DICTS>
  </SPELL_ENGINE>
</SPELL_CONFIG>
```

## Stemming.xml elements

The Stemming.xml file contains an element for each language that is enabled for stemming.

The stemming feature allows the system to consider alternate forms of individual words as equivalent for the purpose of search query matching. For example, it is often desirable for singular nouns to match their plural equivalents in the searchable text, and vice versa. See "Using Stemming and Thesaurus" in the *Endeca MDEX Engine Development Guide* for more information.

### STEM\_CS

Not supported.

### STEM\_DA

Not supported.

### STEM\_DE

A STEM\_DE element specifies whether German is enabled for stemming.

**DTD**

```
<!ELEMENT STEM_DE EMPTY>
<!ATTLIST STEM_DE ENABLE (TRUE|FALSE) "FALSE">
<!ATTLIST STEM_DE USE_STATIC_WORDFORMS (TRUE|FALSE) "TRUE">
<!ATTLIST STEM_DE USE_COMPOUND_MATCHING (TRUE|FALSE) "FALSE">
```

**Attributes**

The following sections describe the STEM\_DE element's attributes.

**ENABLE**

Set to TRUE if the language is enabled for stemming. If the attribute is set to FALSE or if the language's element is not listed in Stemming.xml, then the language is not enabled for stemming.

**USE\_STATIC\_WORDFORMS**

If set to TRUE, the stemming feature uses the static word form files shipped with the MDEX Engine package. If set to FALSE, the Indexer generates the word forms file from the corpus, using the dynamic stemming add-in.

**USE\_COMPOUND\_MATCHING**

If set to TRUE, the stemming feature is extended to allow matching query words that are subsets of indexed compound words.

**Sub-elements**

The STEM\_DE element has no sub-elements.

**Example**

This example enables German stemming, enables the use of the static word form files from the MDEX Engine package, and disables the matching of query words that are subsets of indexed compound words.

```
<STEMMING>
  <STEM_DE ENABLE="TRUE" />
  <STEM_DE USE_STATIC_WORDFORMS="TRUE" />
  <STEM_DE USE_COMPOUND_MATCHING="FALSE" />
</STEMMING>
```

**STEM\_EL**

Not supported.

**STEM\_EN\_UK**

Not supported.

**STEM\_EN\_US**

A STEM\_EN\_US element specifies whether U.S. English is enabled for stemming.

**DTD**

```
<!ELEMENT STEM_EN_US EMPTY>
<!ATTLIST STEM_EN_US ENABLE (TRUE|FALSE) "TRUE">
```

```
<!ATTLIST STEM_EN_US USE_STATIC_WORDFORMS (TRUE|FALSE) "TRUE">
<!ATTLIST STEM_EN_US USE_COMPOUND_MATCHING (TRUE|FALSE) "FALSE">
```

### Attributes

The following sections describe the STEM\_EN\_US element's attributes.

#### ENABLE

Set to TRUE if the language is enabled for stemming. If the attribute is set to FALSE or if the language's element is not listed in Stemming.xml, then the language is not enabled for stemming.

#### USE\_STATIC\_WORDFORMS

If set to TRUE, the stemming feature uses the static word form files shipped with the MDEX Engine package. If set to FALSE, the Indexer generates the word forms file from the corpus, using the dynamic stemming add-in.

#### USE\_COMPOUND\_MATCHING

If set to TRUE, the stemming feature is extended to allow matching query words that are subsets of indexed compound words.

### Sub-elements

The STEM\_EN\_US element has no sub-elements.

### Example

This example enables U.S. English stemming, enables the use of the static word form files from the MDEX Engine package, and disables the matching of query words that are subsets of indexed compound words.

```
<STEMMING>
  <STEM_EN_US ENABLE="TRUE" />
  <STEM_EN_US USE_STATIC_WORDFORMS="TRUE" />
  <STEM_EN_US USE_COMPOUND_MATCHING="FALSE" />
</STEMMING>
```

## STEM\_ES

A STEM\_ES element specifies whether Spanish is enabled for stemming.

### DTD

```
<!ELEMENT STEM_ES EMPTY>
<!ATTLIST STEM_ES ENABLE (TRUE|FALSE) "FALSE">
<!ATTLIST STEM_ES USE_STATIC_WORDFORMS (TRUE|FALSE) "TRUE">
<!ATTLIST STEM_ES USE_COMPOUND_MATCHING (TRUE|FALSE) "FALSE">
```

### Attributes

The following sections describe the STEM\_ES element's attributes.

#### ENABLE

Set to TRUE if the language is enabled for stemming. If the attribute is set to FALSE or if the language's element is not listed in Stemming.xml, then the language is not enabled for stemming.

#### USE\_STATIC\_WORDFORMS

If set to TRUE, the stemming feature uses the static word form files shipped with the MDEX Engine package. If set to FALSE, the Indexer generates the word forms file from the corpus, using the dynamic stemming add-in.

## USE\_COMPOUND\_MATCHING

If set to TRUE, the stemming feature is extended to allow matching query words that are subsets of indexed compound words.

### Sub-elements

The STEM\_ES element has no sub-elements.

### Example

This example enables Spanish stemming, enables the use of the static word form files from the MDEX Engine package, and disables the matching of query words that are subsets of indexed compound words.

```

<STEMMING>
  <STEM_ES ENABLE="TRUE" />
  <STEM_ES USE_STATIC_WORDFORMS="TRUE" />
  <STEM_ES USE_COMPOUND_MATCHING="FALSE" />
</STEMMING>

```

## STEM\_FR

A STEM\_FR element specifies whether French is enabled for stemming.

### DTD

```

<!ELEMENT STEM_FR EMPTY>
<!ATTLIST STEM_FR ENABLE (TRUE|FALSE) "FALSE">
<!ATTLIST STEM_FR USE_STATIC_WORDFORMS (TRUE|FALSE) "TRUE">
<!ATTLIST STEM_FR USE_COMPOUND_MATCHING (TRUE|FALSE) "FALSE">

```

### Attributes

The following sections describe the STEM\_FR element's attributes.

#### ENABLE

Set to TRUE if the language is enabled for stemming. If the attribute is set to FALSE or if the language's element is not listed in Stemming.xml, then the language is not enabled for stemming.

#### USE\_STATIC\_WORDFORMS

If set to TRUE, the stemming feature uses the static word form files shipped with the MDEX Engine package. If set to FALSE, the Indexer generates the word forms file from the corpus, using the dynamic stemming add-in.

#### USE\_COMPOUND\_MATCHING

If set to TRUE, the stemming feature is extended to allow matching query words that are subsets of indexed compound words.

### Sub-elements

The STEM\_FR element has no sub-elements.

### Example

This example enables French stemming, enables the use of the static word form files from the MDEX Engine package, and disables the matching of query words that are subsets of indexed compound words.

```
<STEMMING>
  <STEM_FR ENABLE="TRUE" />
  <STEM_FR USE_STATIC_WORDFORMS="TRUE" />
  <STEM_FR USE_COMPOUND_MATCHING="FALSE" />
</STEMMING>
```

## STEM\_HU

Not supported.

## STEM\_IT

A STEM\_IT element specifies whether Italian is enabled for stemming.

### DTD

```
<!ELEMENT STEM_IT EMPTY>
<!ATTLIST STEM_IT ENABLE (TRUE | FALSE) "FALSE">
<!ATTLIST STEM_IT USE_STATIC_WORDFORMS (TRUE | FALSE) "TRUE">
<!ATTLIST STEM_IT USE_COMPOUND_MATCHING (TRUE | FALSE) "FALSE">
```

### Attributes

The following sections describe the STEM\_IT element's attributes.

#### ENABLE

Set to TRUE if the language is enabled for stemming. If the attribute is set to FALSE or if the language's element is not listed in Stemming.xml, then the language is not enabled for stemming.

#### USE\_STATIC\_WORDFORMS

If set to TRUE, the stemming feature uses the static word form files shipped with the MDEX Engine package. If set to FALSE, the Indexer generates the word forms file from the corpus, using the dynamic stemming add-in.

#### USE\_COMPOUND\_MATCHING

If set to TRUE, the stemming feature is extended to allow matching query words that are subsets of indexed compound words.

### Sub-elements

The STEM\_IT element has no sub-elements.

### Example

This example enables Italian stemming, enables the use of the static word form files from the MDEX Engine package, and disables the matching of query words that are subsets of indexed compound words.

```
<STEMMING>
  <STEM_IT ENABLE="TRUE" />
  <STEM_IT USE_STATIC_WORDFORMS="TRUE" />
  <STEM_IT USE_COMPOUND_MATCHING="FALSE" />
</STEMMING>
```

## STEM\_JP

A STEM\_JP element specifies whether Japanese is enabled for stemming.

### DTD

```
<!ELEMENT STEM_JP EMPTY>
<!ATTLIST STEM_JP ENABLE (TRUE|FALSE) "FALSE">
<!ATTLIST STEM_JP USE_STATIC_WORDFORMS (TRUE|FALSE) "FALSE">
<!ATTLIST STEM_JP USE_COMPOUND_MATCHING (TRUE|FALSE) "FALSE">
```

### Attributes

The following sections describe the STEM\_JP element's attributes.

#### ENABLE

Set to TRUE if the language is enabled for stemming. If the attribute is set to FALSE or if the language's element is not listed in Stemming.xml, then the language is not enabled for stemming.

#### USE\_STATIC\_WORDFORMS

If set to TRUE, the stemming feature uses the static word form files shipped with the MDEX Engine package. If set to FALSE, the Indexer generates the word forms file from the corpus, using the dynamic stemming add-in.

#### USE\_COMPOUND\_MATCHING

If set to TRUE, the stemming feature is extended to allow matching query words that are subsets of indexed compound words.

### Sub-elements

The STEM\_JP element has no sub-elements.

### Example

This example enables Japanese stemming, enables the use of the static word form files from the MDEX Engine package, and disables the matching of query words that are subsets of indexed compound words.

```
<STEMMING>
  <STEM_JP ENABLE="TRUE" />
  <STEM_JP USE_STATIC_WORDFORMS="TRUE" />
  <STEM_JP USE_COMPOUND_MATCHING="FALSE" />
</STEMMING>
```

## STEM\_KO

A STEM\_KO element specifies whether Korean is enabled for stemming.

### DTD

```
<!ELEMENT STEM_KO EMPTY>
<!ATTLIST STEM_KO ENABLE (TRUE|FALSE) "FALSE">
<!ATTLIST STEM_KO USE_STATIC_WORDFORMS (TRUE|FALSE) "FALSE">
<!ATTLIST STEM_KO USE_COMPOUND_MATCHING (TRUE|FALSE) "FALSE">
```

### Attributes

The following sections describe the STEM\_KO element's attributes.

**ENABLE**

Set to TRUE if the language is enabled for stemming. If the attribute is set to FALSE or if the language's element is not listed in Stemming.xml, then the language is not enabled for stemming.

**USE\_STATIC\_WORDFORMS**

If set to TRUE, the stemming feature uses the static word form files shipped with the MDEX Engine package. If set to FALSE, the Indexer generates the word forms file from the corpus, using the dynamic stemming add-in.

**USE\_COMPOUND\_MATCHING**

If set to TRUE, the stemming feature is extended to allow matching query words that are subsets of indexed compound words.

**Sub-elements**

The STEM\_KO element has no sub-elements.

**Example**

This example enables Korean stemming, enables the use of the static word form files from the MDEX Engine package, and disables the matching of query words that are subsets of indexed compound words.

```
<STEMMING>
  <STEM_KO ENABLE="TRUE" />
  <STEM_KO USE_STATIC_WORDFORMS="TRUE" />
  <STEM_KO USE_COMPOUND_MATCHING="FALSE" />
</STEMMING>
```

**STEM\_NL**

A STEM\_NL element specifies whether Dutch is enabled for stemming.

**DTD**

```
<!ELEMENT STEM_NL EMPTY>
<!ATTLIST STEM_NL ENABLE (TRUE|FALSE) "FALSE">
<!ATTLIST STEM_NL USE_STATIC_WORDFORMS (TRUE|FALSE) "TRUE">
<!ATTLIST STEM_NL USE_COMPOUND_MATCHING (TRUE|FALSE) "FALSE">
```

**Attributes**

The following sections describe the STEM\_NL element's attributes.

**ENABLE**

Set to TRUE if the language is enabled for stemming. If the attribute is set to FALSE or if the language's element is not listed in Stemming.xml, then the language is not enabled for stemming.

**USE\_STATIC\_WORDFORMS**

If set to TRUE, the stemming feature uses the static word form files shipped with the MDEX Engine package. If set to FALSE, the Indexer generates the word forms file from the corpus, using the dynamic stemming add-in.

**USE\_COMPOUND\_MATCHING**

If set to TRUE, the stemming feature is extended to allow matching query words that are subsets of indexed compound words.

### Sub-elements

The STEM\_NL element has no sub-elements.

### Example

This example enables Dutch stemming, enables the use of the static word form files from the MDEX Engine package, and disables the matching of query words that are subsets of indexed compound words.

```

<STEMMING>
  <STEM_NL ENABLE="TRUE" />
  <STEM_NL USE_STATIC_WORDFORMS="TRUE" />
  <STEM_NL USE_COMPOUND_MATCHING="FALSE" />
</STEMMING>

```

## STEM\_PL

Not supported.

## STEM\_PT

A STEM\_PT element specifies whether Portuguese is enabled for stemming.

### DTD

```

<!ELEMENT STEM_PT EMPTY>
<!ATTLIST STEM_PT ENABLE (TRUE|FALSE) "FALSE">
<!ATTLIST STEM_PT USE_STATIC_WORDFORMS (TRUE|FALSE) "TRUE">
<!ATTLIST STEM_PT USE_COMPOUND_MATCHING (TRUE|FALSE) "FALSE">

```

### Attributes

The following sections describe the STEM\_PT element's attributes.

#### ENABLE

Set to TRUE if the language is enabled for stemming. If the attribute is set to FALSE or if the language's element is not listed in Stemming.xml, then the language is not enabled for stemming.

#### USE\_STATIC\_WORDFORMS

If set to TRUE, the stemming feature uses the static word form files shipped with the MDEX Engine package. If set to FALSE, the Indexer generates the word forms file from the corpus, using the dynamic stemming add-in.

#### USE\_COMPOUND\_MATCHING

If set to TRUE, the stemming feature is extended to allow matching query words that are subsets of indexed compound words.

### Sub-elements

The STEM\_PT element has no sub-elements.

### Example

This example enables Portuguese stemming, enables the use of the static word form files from the MDEX Engine package, and disables the matching of query words that are subsets of indexed compound words.

```
<STEMMING>
  <STEM_PT ENABLE="TRUE" />
  <STEM_PT USE_STATIC_WORDFORMS="TRUE" />
  <STEM_PT USE_COMPOUND_MATCHING="FALSE" />
</STEMMING>
```

## STEM\_XX

Not supported.

## STEM\_ZH\_CN

A STEM\_ZH\_CN element specifies whether Chinese (simplified) is enabled for stemming.

### DTD

```
<!ELEMENT STEM_ZH_CN EMPTY>
<!ATTLIST STEM_ZH_CN ENABLE (TRUE|FALSE) "FALSE">
<!ATTLIST STEM_ZH_CN USE_STATIC_WORDFORMS (TRUE|FALSE) "FALSE">
<!ATTLIST STEM_ZH_CN USE_COMPOUND_MATCHING (TRUE|FALSE) "FALSE">
```

### Attributes

The following sections describe the STEM\_ZH\_CN element's attributes.

#### ENABLE

Set to TRUE if the language is enabled for stemming. If the attribute is set to FALSE or if the language's element is not listed in Stemming.xml, then the language is not enabled for stemming.

#### USE\_STATIC\_WORDFORMS

If set to TRUE, the stemming feature uses the static word form files shipped with the MDEX Engine package. If set to FALSE, the Indexer generates the word forms file from the corpus, using the dynamic stemming add-in.

#### USE\_COMPOUND\_MATCHING

If set to TRUE, the stemming feature is extended to allow matching query words that are subsets of indexed compound words.

### Sub-elements

The STEM\_ZH\_CN element has no sub-elements.

### Example

This example enables Chinese (simplified) stemming, enables the use of the static word form files from the MDEX Engine package, and disables the matching of query words that are subsets of indexed compound words.

```
<STEMMING>
  <STEM_ZH_CN ENABLE="TRUE" />
  <STEM_ZH_CN USE_STATIC_WORDFORMS="TRUE" />
```

```
<STEM_ZH_CN USE_COMPOUND_MATCHING="FALSE" />
</STEMMING>
```

## STEM\_ZH\_TW

A STEM\_ZH\_TW element specifies whether Chinese (traditional) is enabled for stemming.

### DTD

```
<!ELEMENT STEM_ZH_TW EMPTY>
<!ATTLIST STEM_ZH_TW ENABLE (TRUE|FALSE) "FALSE">
<!ATTLIST STEM_ZH_TW USE_STATIC_WORDFORMS (TRUE|FALSE) "FALSE">
<!ATTLIST STEM_ZH_TW USE_COMPOUND_MATCHING (TRUE|FALSE) "FALSE">
```

### Attributes

The following sections describe the STEM\_ZH\_TW element's attributes.

#### ENABLE

Set to TRUE if the language is enabled for stemming. If the attribute is set to FALSE or if the language's element is not listed in Stemming.xml, then the language is not enabled for stemming.

#### USE\_STATIC\_WORDFORMS

If set to TRUE, the stemming feature uses the static word form files shipped with the MDEX Engine package. If set to FALSE, the Indexer generates the word forms file from the corpus, using the dynamic stemming add-in.

#### USE\_COMPOUND\_MATCHING

If set to TRUE, the stemming feature is extended to allow matching query words that are subsets of indexed compound words.

### Sub-elements

The STEM\_ZH\_TW element has no sub-elements.

### Example

This example enables Chinese (traditional) stemming, enables the use of the static word form files from the MDEX Engine package, and disables the matching of query words that are subsets of indexed compound words.

```
<STEMMING>
  <STEM_ZH_TW ENABLE="TRUE" />
  <STEM_ZH_TW USE_STATIC_WORDFORMS="TRUE" />
  <STEM_ZH_TW USE_COMPOUND_MATCHING="FALSE" />
</STEMMING>
```

## Stop\_words.xml elements

The Stop\_words.xml file contains words that should be eliminated from a query before it is processed by the MDEX Engine.

Each stop is specified in a STOP\_WORD element.

## STOP\_WORD

The STOP\_WORD element identifies words that should be eliminated from a query before it is processed. Examples of common stop words include the words "the" and "of".

### DTD

```
<!ELEMENT STOP_WORD (#PCDATA)>
```

### Attributes

The STOP\_WORD element has no attributes.

### Sub-elements

The STOP\_WORD element has no sub-elements.

### Example

This example shows a common set of stop words.

```
<STOP_WORDS>
  <STOP_WORD>a</STOP_WORD>
  <STOP_WORD>an</STOP_WORD>
  <STOP_WORD>of</STOP_WORD>
  <STOP_WORD>the</STOP_WORD>
</STOP_WORDS>
```

## STOP\_WORDS

A STOP\_WORDS element specifies the stop words enabled in your application.

Each stop word is represented by a STOP\_WORD element.

### DTD

```
<!ELEMENT STOP_WORDS
  ( COMMENT?
    , STOP_WORD*
  )
>
```

### Attributes

The STOP\_WORDS element has no attributes.

### Sub-elements

The following table provides a brief overview of the STOP\_WORDS sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.

Sub-element	Brief description
STOP_WORD	Identifies words that should be eliminated from a query before it is processed.

### Example

This example shows a common set of stop words.

```
<STOP_WORDS>
  <STOP_WORD>a</STOP_WORD>
  <STOP_WORD>an</STOP_WORD>
  <STOP_WORD>of</STOP_WORD>
  <STOP_WORD>the</STOP_WORD>
</STOP_WORDS>
```

## Studio\_project.esp elements

The Studio\_project.esp file contains the elements for a Developer Studio project.

A Developer Studio project has a project file that is typically the name of your project followed by an Endeca Studio Project (.esp) extension. This file tracks the other files used in the project.

### PROJECT\_FILE

A PROJECT\_FILE element specifies an individual project file of a Developer Studio project.

All the project files in a project constitute the instance configuration. If a project file is missing, Developer Studio throws an error when attempting to load the project.

#### DTD

```
<!ELEMENT PROJECT_FILE EMPTY>
<!ATTLIST PROJECT_FILE
  TYPE          CDATA   #REQUIRED
  %url;
  HIGHEST_ID    CDATA   #IMPLIED
>
```

#### Attributes

The following sections describe the PROJECT\_FILE element's attributes.

##### TYPE

Specifies the root element for the project file.

##### URL

Specifies the file name element for the project file.

##### HIGHEST\_ID

Identifies the highest rule group ID number that has ever been used in your project's defined rule groups.

### Sub-elements

The PROJECT\_FILE element has no sub-elements.

### Example

This example shows three of the PROJECT\_FILE elements in the wine demo project file.

```

<STUDIO_PROJECT APP_VERSION="5.1" NAME="Sample Wine Pipeline" VERSION="500">
  <PROJECT_MANAGER_CONFIG APPLICATION="wine" FORGE_OP_CONFIG="--pruneAutoGen"
    PORT="8888" URL="localhost"/>
  <PROJECT_PIPELINE URL="pipeline.epx"/>
  <PROJECT_FILE TYPE="ANALYTICS_CONFIG" URL="wine.analytics_config.xml"/>
  <PROJECT_FILE TYPE="DERIVED_PROPS" URL="wine.derived_props.xml"/>
  <PROJECT_FILE TYPE="DIMENSION_GROUPS" URL="wine.dimension_groups.xml"/>
  . . .
</STUDIO_PROJECT>

```

## PROJECT\_MANAGER\_CONFIG

A PROJECT\_MANAGER\_CONFIG element specifies the host and port of Endeca Workbench and application name this project is associated with.

### DTD

```

<!ELEMENT PROJECT_MANAGER_CONFIG EMPTY>
<!ATTLIST PROJECT_MANAGER_CONFIG
  %url;
  PORT          CDATA   #REQUIRED
  APPLICATION   CDATA   #IMPLIED
  FORGE_OP_CONFIG CDATA  #IMPLIED
  ENE_OP_CONFIG CDATA  #IMPLIED
  AENE_OP_CONFIG CDATA  #IMPLIED
  INDEXER_OP_CONFIG CDATA #IMPLIED
>

```

### Attributes

The following sections describe the PROJECT\_MANAGER\_CONFIG element's attributes. Note that the FORGE\_OP\_CONFIG, ENE\_OP\_CONFIG, AENE\_OP\_CONFIG, and INDEXER\_OP\_CONFIG attributes are no longer used.

#### URL

Specifies the host on which Endeca Workbench is running.

#### PORT

Specifies the port number on which on Endeca Workbench listens.

#### APPLICATION

Specifies the application in Endeca Workbench that the Developer Studio project is associated with.

### Sub-elements

The PROJECT\_MANAGER\_CONFIG element has no sub-elements.

## Example

This example shows some elements of the wine demo project file.

```
<STUDIO_PROJECT APP_VERSION="6.0"
  NAME="Sample Wine Pipeline" VERSION="600">
  <PROJECT_MANAGER_CONFIG APPLICATION="wine" PORT="8888"
    URL="localhost"/>
  <PROJECT_PIPELINE URL="pipeline.epx"/>
  <PROJECT_FILE TYPE="ANALYTICS_CONFIG" URL="wine.analytics_config.xml"/>
  <PROJECT_FILE TYPE="DERIVED_PROPS" URL="wine.derived_props.xml"/>
  <PROJECT_FILE TYPE="DIMENSION_GROUPS" URL="wine.dimension_groups.xml"/>
  ...
</STUDIO_PROJECT>
```

## PROJECT\_PIPELINE

A PROJECT\_PIPELINE element specifies the name of the project's pipeline.epx file.

### DTD

```
<!ELEMENT PROJECT_PIPELINE EMPTY>
<!ATTLIST PROJECT_PIPELINE
  %url;
  TYPE CDATA #IMPLIED
>
```

### Attributes

The following sections describe the PROJECT\_PIPELINE element's attributes.

#### URL

Specifies the file name element for the pipeline.epx file

#### TYPE

Specifies the root element for the pipeline.epx file.

### Sub-elements

The PROJECT\_PIPELINE element has no sub-elements.

### Example

This example shows the PROJECT\_PIPELINE element of the wine demo project file.

```
<STUDIO_PROJECT APP_VERSION="6.0"
  NAME="Sample Wine Pipeline" VERSION="600">
  <PROJECT_MANAGER_CONFIG APPLICATION="wine" PORT="8888"
    URL="localhost"/>
  <PROJECT_PIPELINE URL="pipeline.epx"/>
  <PROJECT_FILE TYPE="ANALYTICS_CONFIG" URL="wine.analytics_config.xml"/>
  ...
</STUDIO_PROJECT>
```

## STUDIO\_PROJECT

A STUDIO\_PROJECT element specifies the name of a project and version information for Developer Studio.

**DTD**

```

<!ELEMENT STUDIO_PROJECT
  ( COMMENT?
  , PROJECT_MANAGER_CONFIG
  , PROJECT_RESOURCE_CONFIG?
  , PROJECT_PIPELINE+
  , PROJECT_NAV_CONFIG?
  , PROJECT_FILE*
  )
>
<!ATTLIST STUDIO_PROJECT
  NAME          CDATA      #REQUIRED
  VERSION       CDATA      #IMPLIED
  APP_VERSION   CDATA      #IMPLIED
  TYPE          CDATA      #IMPLIED
>

```

**Attributes**

The following sections describe the STUDIO\_PROJECT element's attributes.

**NAME**

Specifies the project name.

**VERSION**

Specifies the version of Developer Studio used to create the project.

**APP\_VERSION**

Specifies the version of Developer Studio used to create the project.

**Sub-elements**

The following table provides a brief overview of the STUDIO\_PROJECT sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.
PROJECT_MANAGER_CONFIG	Specifies the host, port, and command line options that Developer Studio passes to Endeca Workbench.
PROJECT_RESOURCE_CONFIG	Deprecated.
PROJECT_PIPELINE	Specifies the name of the project's pipeline.epx file.
PROJECT_NAV_CONFIG	Deprecated.
PROJECT_FILE	Specifies an individual project file of a Developer Studio project.

**Example**

This example shows some of the sub-elements contain in the STUDIO\_PROJECT element of the wine demo project file.

```

<STUDIO_PROJECT APP_VERSION="6.0"
  NAME="Sample Wine Pipeline" VERSION="600">
  <PROJECT_MANAGER_CONFIG APPLICATION="wine" PORT="8888"

```

```

    URL="localhost" />
<PROJECT_PIPELINE URL="pipeline.epx" />
<PROJECT_FILE TYPE="ANALYTICS_CONFIG" URL="wine.analytics_config.xml" />
<PROJECT_FILE TYPE="DERIVED_PROPS" URL="wine.derived_props.xml" />
<PROJECT_FILE TYPE="DIMENSION_GROUPS" URL="wine.dimension_groups.xml" />
    . . .
</STUDIO_PROJECT>

```

## Thesaurus.xml elements

The Thesaurus.xml file contains thesaurus entries for your application.

Thesaurus entries provide a means to account for alternate forms of a user's query. These entries provide concept-level mappings between words and phrases. See "Using Stemming and Thesaurus" in the *Endeca MDEX Engine Development Guide* for more information.

## THESAURUS

A THESAURUS element contains the term equivalence mappings for an application.

THESAURUS is the root element for all thesaurus entries.

Note that the order of sub-elements within THESAURUS is significant. You should add sub-elements in the order in which they are listed in the DTD section. For example, THESAURUS\_ENTRY sub-elements appear before THESAURUS\_ENTRY\_ONEWAY. See the example below.

### DTD

```

<!ELEMENT THESAURUS
  ( COMMENT?
    , THESAURUS_ENTRY*
    , THESAURUS_ENTRY_ONEWAY*
  )
>

```

### Attributes

The THESAURUS element has no attributes.

### Sub-elements

The following table provides a brief overview of the THESAURUS sub-elements.

Sub-element	Brief description
COMMENT	Associates a comment with a parent element and preserves the comment when the file is rewritten. This element provides an alternative to using inline XML comments of the form <!-- ... -->.
THESAURUS_ENTRY	Indicates a set of word forms (contained in THESAURUS_FORM elements) that are equivalent.
THESAURUS_ENTRY_ONEWAY	Specifies single-direction equivalency mappings.

## Example

This example shows the thesaurus entries for an application.

```

<THESAURUS>
  <THESAURUS_ENTRY>
    <THESAURUS_FORM>france</THESAURUS_FORM>
    <THESAURUS_FORM>french</THESAURUS_FORM>
  </THESAURUS_ENTRY>
  <THESAURUS_ENTRY_ONEWAY>
    <THESAURUS_FORM_FROM>Red wine</THESAURUS_FORM_FROM>
    <THESAURUS_FORM_TO>Merlot</THESAURUS_FORM_TO>
    <THESAURUS_FORM_TO>Shiraz</THESAURUS_FORM_TO>
    <THESAURUS_FORM_TO>Bordeaux</THESAURUS_FORM_TO>
  </THESAURUS_ENTRY_ONEWAY>
</THESAURUS>

```

## THESAURUS\_ENTRY

The THESAURUS\_ENTRY element indicates a set of word forms that are equivalent.

The word forms are contained in THESAURUS\_FORM elements. A search for any of these forms (including stemming-matched versions) returns hits for all of the forms.

### DTD

```

<!ELEMENT THESAURUS_ENTRY (THESAURUS_FORM+)>

```

### Attributes

The THESAURUS\_ENTRY element has no attributes.

### Sub-elements

The following table provides a brief overview of the THESAURUS\_ENTRY sub-element.

Sub-element	Brief description
THESAURUS_ENTRY	Indicates a set of word forms that are equivalent.

### Example

In this example, the noun and adjective forms of a word are made equivalent.

```

<THESAURUS>
  <THESAURUS_ENTRY>
    <THESAURUS_FORM>france</THESAURUS_FORM>
    <THESAURUS_FORM>french</THESAURUS_FORM>
  </THESAURUS_ENTRY>
</THESAURUS>

```

## THESAURUS\_ENTRY\_ONEWAY

A THESAURUS\_ENTRY\_ONEWAY element specifies a single-direction mapping.

Searches for any of the "from" forms (THESAURUS\_FORM\_FROM elements) also return hits for all of the "to" forms (THESAURUS\_FORM\_TO elements). The other direction is not enabled; that is, searches for the "to" forms do not return results for either the "from" forms or the other "to" forms.

**DTD**

```
<!ELEMENT THESAURUS_ENTRY_ONEWAY
  ( THESAURUS_FORM_FROM
    , THESAURUS_FORM_TO+
  )
>
```

**Attributes**

The THESAURUS\_ENTRY\_ONEWAY element has no attributes.

**Sub-elements**

The following table provides a brief overview of the THESAURUS\_ENTRY\_ONEWAY sub-elements.

Sub-element	Brief description
THESAURUS_FORM_FROM	Specifies the "from" form in a one-way word mapping.
THESAURUS_FORM_TO	Specifies the "to" form in a one-way word mapping.

**Example**

In this example, searches for *Red wine* would return hits for *Red wine* as well as for *Merlot*, *Shiraz*, and *Bordeaux*. Since the equivalence is one-way, more specific searches such as *Shiraz* or *Bordeaux* would not return results for the more general concept *Red wine*.

```
<THESAURUS_ENTRY_ONEWAY>
  <THESAURUS_FORM_FROM>Red wine</THESAURUS_FORM_FROM>
  <THESAURUS_FORM_TO>Merlot</THESAURUS_FORM_TO>
  <THESAURUS_FORM_TO>Shiraz</THESAURUS_FORM_TO>
  <THESAURUS_FORM_TO>Bordeaux</THESAURUS_FORM_TO>
</THESAURUS_ENTRY_ONEWAY>
```

**THESAURUS\_FORM**

The THESAURUS\_FORM element contains a word form that is used by the THESAURUS\_ENTRY element to set an equivalence.

**DTD**

```
<!ELEMENT THESAURUS_FORM (#PCDATA)>
```

**Attributes**

The THESAURUS\_FORM element has no attributes.

**Sub-elements**

The THESAURUS\_FORM element has no sub-elements.

**Example**

In this example, the noun and adjective forms of a word are made equivalent.

```
<THESAURUS>
  <THESAURUS_ENTRY>
    <THESAURUS_FORM>france</THESAURUS_FORM>
```

```
<THESAURUS_FORM>french</THESAURUS_FORM>
</THESAURUS_ENTRY>
</THESAURUS>
```

## THESAURUS\_FORM\_FROM

The THESAURUS\_FORM\_FROM element provides the "from" form within a THESAURUS\_ENTRY\_ONEWAY element.

### DTD

```
<!ELEMENT THESAURUS_FORM_FROM (#PCDATA)>
```

### Attributes

The THESAURUS\_FORM\_FROM element has no attributes.

### Sub-elements

The THESAURUS\_FORM\_FROM element has no sub-elements.

### Example

In this example, searches for `home theater` would return hits for `home theater` as well as for `stereo` and `television`. Because the equivalence is one-way, more specific searches such as `stereo` or `television` would not return results for the more general concept `home theater`.

```
<THESAURUS_ENTRY_ONEWAY>
  <THESAURUS_FORM_FROM>home theater</THESAURUS_FORM_FROM>
  <THESAURUS_FORM_TO>stereo</THESAURUS_FORM_TO>
  <THESAURUS_FORM_TO>television</THESAURUS_FORM_TO>
</THESAURUS_ENTRY_ONEWAY>
```

## THESAURUS\_FORM\_TO

The THESAURUS\_FORM\_TO element provides the "to" form within a THESAURUS\_ENTRY\_ONEWAY element.

### DTD

```
<!ELEMENT THESAURUS_FORM_TO (#PCDATA)>
```

### Attributes

The THESAURUS\_FORM\_TO element has no attributes.

### Sub-elements

The THESAURUS\_FORM\_TO element has no sub-elements.

## Example

In this example, searches for `home theater` would return hits for `home theater` as well as for `stereo` and `television`. Because the equivalence is one-way, more specific searches such as `stereo` or `television` would not return results for the more general concept `home theater`.

```
<THESAURUS_ENTRY_ONEWAY>
  <THESAURUS_FORM_FROM>home theater</THESAURUS_FORM_FROM>
  <THESAURUS_FORM_TO>stereo</THESAURUS_FORM_TO>
  <THESAURUS_FORM_TO>television</THESAURUS_FORM_TO>
</THESAURUS_ENTRY_ONEWAY>
```



# Index

## A

- about this document 11
- analytics\_config.xml
  - about 11
- ASPELL 141
- AUTO\_SUGGEST 135

## B

- BOUND 24

## C

- CLUSTERS 110
- COMMENT 11
- common.dtd
  - COMMENT 11
  - compression 12
- Common.dtd
  - CONFIG 12
  - DIMENSION\_ID 14
  - DIMNAME 14
  - DVAL\_ID 15
  - DVAL\_PATH 15
  - KEY\_DIMENSION 16
  - LOG 17
  - PHRASE 18
  - PROFILE 18
  - PROP 19
  - PROPNAME 19
  - PVAL 20
  - RECORD\_GROUP 20
  - RECORD\_ID 21
  - RECORD\_INDEX 21
- compression 12
- CONFIG 12

## D

- DERIVED\_PROP 22
- DERIVED\_PROPS 23
- Derived\_props.xml
  - about 22
  - DERIVED\_PROP 22
  - DERIVED\_PROPS 23
- DICT\_PER\_LANGUAGE 142
- DID\_YOU\_MEAN 136
- DIMENSION 25
- DIMENSION\_ADAPTER 46
- DIMENSION\_GROUP 32
- DIMENSION\_GROUPS 33

- Dimension\_groups.xml
  - about 31
  - DIMENSION\_GROUP 32
  - DIMENSION\_GROUPS 33
- DIMENSION\_ID 14
- DIMENSION\_IMPORTS 46
- DIMENSION\_INPUTS 95
- DIMENSION\_NODE 26
- DIMENSION\_REF 34
- DIMENSION\_REFS 35
- Dimension\_refs.xml
  - about 33
  - DIMENSION\_REF 34
  - DIMENSION\_REFS 35
- DIMENSION\_SERVER 48
- DIMENSION\_SOURCE 50
- DIMENSIONS 26
- Dimensions.xml
  - about 23
  - BOUND 24
  - DIMENSION 25
  - DIMENSION\_NODE 26
  - DIMENSIONS 26
  - DVAL 27
  - LBOUND 29
  - NUM\_DVALS 29
  - SYN 30
  - UBOUND 31
- DIMNAME 14
- DIMSEARCH\_CONFIG 91
- Dimsearch\_config.xml
  - about 91
  - DIMSEARCH\_CONFIG 91
- DIMSEARCH\_HIERARCHY 35
- DIMSEARCH\_INDEX 36
- Dimsearch\_index.xml
  - about 35
  - DIMSEARCH\_HIERARCHY 35
  - DIMSEARCH\_INDEX 36
- DVAL 27
- DVAL\_ID 15
- DVAL\_ID\_LIMITS 51
- DVAL\_PATH 15
- DVAL\_RANK 38
- DVAL\_RANKS 39
- Dval\_ranks.xml
  - about 38
  - DVAL\_RANK 38
  - DVAL\_RANKS 39
- DVAL\_REF 40
- DVAL\_REFS 41
- Dval\_refs.xml
  - about 40
  - DVAL\_REF 40

Dval\_refs.xml (*continued*)  
 DVAL\_REFS 41  
 DYNAMIC\_RANKING 111

**E**

ESPELL 143  
 EXPRBODY 52  
 EXPRESSION 52  
 EXPRNODE 54  
 externaldimensions.xml, about 42

**F**

FIRST\_DICT 144

**I**

ID\_SERVER 55  
 INDEXER\_ADAPTER 56  
 INPUT 96  
 introduction 11

**J**

JAVA\_MANIPULATOR 58  
 JOIN\_ENTRY 60

**K**

KEY\_DIMENSION 16  
 KEY\_LANGUAGE 42  
 Key\_props.xml, about 42

**L**

LANGUAGES 43  
 Languages.xml  
   about 42  
   KEY\_LANGUAGE 42  
   LANGUAGES 43  
 LBOUND 29  
 LOG 17

**M**

MEMBER\_NAME 136

**N**

NUM\_DVALS 29

**P**

PARTIAL\_MATCH 137  
 PASS\_THROUGH 60  
 PERL\_MANIPULATOR 63

PERL\_METHOD 65  
 PHRASE 18  
 PHRASE\_ADDITIONS 45  
 PHRASES 44  
 Phrases.xml  
   about 44  
   DIMENSION\_IMPORTS 46  
   PHRASE\_ADDITIONS 45  
   PHRASES 44  
 PIPELINE 67  
 Pipeline.epx  
   DVAL\_ID\_LIMITS 51  
   EXPRBODY 52  
 Pipeline.exp  
   about 46  
   DIMENSION\_ADAPTER 46  
   DIMENSION\_SERVER 48  
   DIMENSION\_SOURCE 50  
   EXPRESSION 52  
   EXPRNODE 54  
   ID\_SERVER 55  
   INDEXER\_ADAPTER 56  
   JAVA\_MANIPULATOR 58  
   JOIN\_ENTRY 60  
   PASS\_THROUGH 60  
   PERL\_MANIPULATOR 63  
   PERL\_METHOD 65  
   PIPELINE 67  
   PROP\_MAPPER 69  
   PROP\_MAPPING\_DIM 71  
   PROP\_MAPPING\_NONE 73  
   PROP\_MAPPING\_PROP 73  
   RECORD\_ADAPTER 75  
   RECORD\_ASSEMBLER 81  
   RECORD\_CACHE 82  
   RECORD\_JOIN 83  
   RECORD\_MANIPULATOR 84  
   RECORD\_SOURCE 85  
   ROLLOVER 86  
   TRANSFORMER 88  
   UPDATE\_ADAPTER 89  
 POSITIONAL\_INDEX 133  
 PRECEDENCE\_RULE 92  
 PRECEDENCE\_RULES 93  
 Precedence\_rules.xml  
   about 92  
   PRECEDENCE\_RULE 92  
   PRECEDENCE\_RULES 93  
 PROFILE 18  
 PROFILES 94  
 Profiles.xml  
   about 94  
   PROFILES 94  
 PROJECT 97  
 PROJECT\_FILE 159  
 PROJECT\_MANAGER\_CONFIG 160  
 PROJECT\_PIPELINE 161  
 Project.xml  
   about 95  
   DIMENSION\_INPUTS 95

Project.xml (*continued*)  
 INPUT 96  
 PROJECT 97  
 RECORD\_INPUTS 97  
 PROP 19  
 PROP\_MAPPER 69  
 PROP\_MAPPING\_DIM 71  
 PROP\_MAPPING\_NONE 73  
 PROP\_MAPPING\_PROP 73  
 PROP\_REF 98  
 PROP\_REFS 99  
 Prop\_refs.xml  
 about 98  
 PROP\_REF 98  
 PROP\_REFS 99  
 PROPNAME 19  
 PVAL 20

## R

RECORD\_ADAPTER 75  
 RECORD\_ASSEMBLER 81  
 RECORD\_CACHE 82  
 RECORD\_FILTER 100  
 Record\_filter.xml  
 about 100  
 RECORD\_FILTER 100  
 RECORD\_GROUP 20  
 RECORD\_ID 21  
 RECORD\_ID\_PROP 101  
 Record\_id\_prop.xml  
 about 101  
 RECORD\_ID\_PROP 101  
 RECORD\_INDEX 21  
 RECORD\_INPUTS 97  
 RECORD\_JOIN 83  
 RECORD\_MANIPULATOR 84  
 RECORD\_SORT 102  
 RECORD\_SORT\_CONFIG 103  
 Record\_sort\_config.xml  
 about 102  
 RECORD\_SORT 102  
 RECORD\_SORT\_CONFIG 103  
 RECORD\_SOURCE 85  
 RECORD\_SPEC 104  
 Record\_spec.xml  
 about 103  
 RECORD\_SPEC 104  
 RECSEARCH\_CONFIG 105  
 Recsearch\_config.xml  
 about 104  
 RECSEARCH\_CONFIG 105  
 RECSEARCH\_INDEX 107  
 RECSEARCH\_INDEXES 108  
 Recsearch\_indexes.xml  
 about 106  
 RECSEARCH\_INDEX 107  
 RECSEARCH\_INDEXES 108  
 Refinement\_config.xml  
 about 110  
 CLUSTERS 110  
 DYNAMIC\_RANKING 111  
 REFINEMENTS 113  
 REFINEMENTS\_CONFIG 114  
 STATS 115  
 REFINEMENTS 113  
 REFINEMENTS\_CONFIG 114  
 RELRANK\_APPROXPHRASE 116  
 RELRANK\_EXACT 117  
 RELRANK\_FIELD 117  
 RELRANK\_FIRST 118  
 RELRANK\_FREQ 118, 123  
 RELRANK\_GLOM 119  
 RELRANK\_INTERP 119  
 RELRANK\_MAXFIELD 120  
 RELRANK\_MODULE 120  
 RELRANK\_NTERMS 121  
 RELRANK\_NUMFIELDS 122  
 RELRANK\_PHRASE 122  
 RELRANK\_SPELL 123  
 RELRANK\_STATIC 124  
 RELRANK\_STRATEGIES 125  
 Relrank\_strategies.xml  
 about 116  
 RELRANK\_APPROXPHRASE 116  
 RELRANK\_EXACT 117  
 RELRANK\_FIELD 117  
 RELRANK\_FIRST 118  
 RELRANK\_FREQ 118, 123  
 RELRANK\_GLOM 119  
 RELRANK\_INTERP 119  
 RELRANK\_MAXFIELD 120  
 RELRANK\_MODULE 120  
 RELRANK\_NTERMS 121  
 RELRANK\_NUMFIELDS 122  
 RELRANK\_PHRASE 122  
 RELRANK\_SPELL 123  
 RELRANK\_STATIC 124  
 RELRANK\_STRATEGIES 125  
 RELRANK\_STRATEGY 125  
 RELRANK\_WFREQ 127  
 RELRANK\_STRATEGY 125  
 RELRANK\_WFREQ 127  
 RENDER 128  
 RENDER\_CONFIG 129  
 Render\_config.xml  
 about 128  
 RENDER 128  
 RENDER\_CONFIG 129  
 ROLLOVER 86  
 ROLLUP 130  
 ROLLUPS 131  
 Rollups.xml  
 about 130  
 ROLLUP 130  
 ROLLUPS 131

**S**

SEARCH\_CHAR 131  
 SEARCH\_CHARS 132  
 Search\_chars.xml  
   about 131  
   SEARCH\_CHAR 131  
   SEARCH\_CHARS 132  
 SEARCH\_CORRECTION\_PARAMETERS 138  
 SEARCH\_INDEX 133  
 SEARCH\_INTERFACE 139  
 Search\_interface.xml  
   about 135  
   AUTO\_SUGGEST 135  
   DID\_YOU\_MEAN 136  
   MEMBER\_NAME 136  
   PARTIAL\_MATCH 137  
   SEARCH\_CORRECTION\_PARAMETERS 138  
   SEARCH\_INTERFACE 139  
 Searchindex.xml  
   about 132  
   POSITIONAL\_INDEX 133  
   SEARCH\_INDEX 133  
   WILDCARD\_INDEX 134  
 SPELL\_CONFIG 145  
 Spell\_config.exp  
   about 141  
 Spell\_config.xml  
   ASPELL 141  
   DICT\_PER\_LANGUAGE 142  
   ESPELL 143  
   FIRST\_DICT 144  
   SPELL\_CONFIG 145  
   SPELL\_ENGINE 146  
   UNION\_DICTS 147  
 SPELL\_ENGINE 146  
 STATS 115  
 STEM\_DE 149  
 STEM\_EN\_US 150  
 STEM\_ES 150  
 STEM\_FR 151  
 STEM\_IT 152  
 STEM\_JP 153  
 STEM\_KO 153  
 STEM\_NL 154  
 STEM\_PT 155  
 STEM\_ZH\_CN 156  
 STEM\_ZH\_TW 157  
 Stemming.xml  
   about 148  
   STEM\_DE 149  
   STEM\_EN\_US 150  
   STEM\_ES 150  
   STEM\_FR 151

Stemming.xml (*continued*)

STEM\_IT 152  
 STEM\_JP 153  
 STEM\_KO 153  
 STEM\_NL 154  
 STEM\_PT 155  
 STEM\_ZH\_CN 156  
 STEM\_ZH\_TW 157  
 STOP\_WORD 158  
 STOP\_WORDS 158  
 Stop\_words.xml  
   about 157  
   STOP\_WORD 158  
   STOP\_WORDS 158  
 STUDIO\_PROJECT 162  
 Studio\_project.esp  
   PROJECT\_FILE 159  
   PROJECT\_MANAGER\_CONFIG 160  
   PROJECT\_PIPELINE 161  
   STUDIO\_PROJECT 162  
 Studio\_project.xml  
   about 159  
 SYN 30

**T**

THESAURUS 163  
 THESAURUS\_ENTRY 164  
 THESAURUS\_ENTRY\_ONEWAY 164  
 THESAURUS\_FORM 165  
 THESAURUS\_FORM\_FROM 166  
 THESAURUS\_FORM\_TO 166  
 Thesaurus.xml  
   about 163  
   THESAURUS 163  
   THESAURUS\_ENTRY 164  
   THESAURUS\_ENTRY\_ONEWAY 164  
   THESAURUS\_FORM 165  
   THESAURUS\_FORM\_FROM 166  
   THESAURUS\_FORM\_TO 166  
 TRANSFORMER 88

**U**

UBOUND 31  
 UNION\_DICTS 147  
 UPDATE\_ADAPTER 89

**W**

welcome 11  
 WILDCARD\_INDEX 134