Configuring Oracle® Java CAPS Master Indexes (Repository)

Part No: 821–2558
January 2011

ORACLE

120126@25097

Contents

Configuring Oracle Java CAPS Master Indexes (Repository)

The topics listed here provide information about configuring a master index application. You can perform the tasks listed on this page after you create the master index framework. For more in-depth information about the concepts behind configuring a master index application, see *Understanding Oracle Java CAPS Master Index Configuration Options (Repository)*.

Note that Java CAPS includes two versions of Oracle Java CAPS Master Index. Oracle Java CAPS Master Index (Repository) is installed in the Java CAPS repository and provides all the functionality of previous versions in the new Java CAPS environment. Oracle Java CAPS Master Index is a service-enabled version of the master index that is installed directly into NetBeans. It includes all of the features of Oracle Java CAPS Master Index (Repository) plus several new features, like data analysis, data cleansing, data loading, and an improved Data Manager GUI. Both products are components of the Oracle Java CAPS Master Data Management (MDM) Suite. This document relates to Oracle Java CAPS Master Index (Repository) only.

What You Need to Know

These topics provide information you should to know before you start customizing a master index application.

What You Need to Do

These topics provide instructions on how to configure the components of a master index, either by modifying the XML files directly or by using the graphical Configuration Editor.

More Information

These topics provide additional information you should know when configuring a master index application.

Related Topics

t

Several topics provide information and instructions for implementing and using a Repository-based master index application. For a complete list of topics related to working with Oracle Java CAPS Master Index (Repository), see "Related Topics" in *Developing Oracle Java CAPS Master Indexes (Repository)*.

Master Index Configuration Overview (Repository)

Oracle Java CAPS Master Index provides a very flexible framework for creating a master index application that is customized for your requirements. A Oracle Java CAPS Master Index project includes several files in XML format that define the configuration of the runtime environment. You can configure a master index application by modifying the XML files directly or by using the Configuration Editor. Make sure to verify the configuration of the application before deploying the project.

Note – It is helpful to review the information provided in *Understanding Oracle Java CAPS Master Index Configuration Options (Repository)* to learn about the relationships between the files and what components and processes can be configured. Certain components can only be configured by modifying the XML files directly.

The following topics provide an overview of the configuration files and editors.

About the Master Index Configuration Files (Repository)

Several XML configuration files define primary characteristics of the master index application, such as how data is processed, queried, and matched. These files configure runtime components of the master index application.

Object Definition

In the wizard, you define the objects and fields contained in the object structure, along with properties for those fields. The information you specify is written to the Object Definition file in the master index project. This file defines the objects stored in the master index application and their relationships to one another. It also defines the fields contained in each object, as well as certain properties of each field, such as length, data type, whether it is required, whether it is a unique key, and so on. This file contains one parent object; all other objects must be child objects to that parent object. The object structure you define in the Object Definition file determines the structure of the database tables that store object data, the structure of the Java API, and the structure of the OTD generated for the project.

Candidate Select

In the Candidate Select file, you configure the *Query Builder* component of the master index application and define the available queries. In this file, you define the types of queries that can be performed from the Enterprise Data Manager (EDM) and the queries that are used during the match process. You can define both phonetic and alphanumeric searches for the EDM. By default, these are called *basic queries*. You can also define *blocking queries*, which define blocks of criteria fields for the match process. The master index application queries the database using the criteria defined in each block, one at a time. After completing a query on the criteria defined in one block, it performs another pass using the next block of defined criteria. Blocking queries can also be used in place of the basic phonetic query in the EDM.

Match Field

In the Match Field file, you configure the *Matching Service* by specifying the fields to be standardized and the fields to be used for matching, as well as defining how the fields are standardized and matched. This file specifies the match and standardization engines to use and the query process for matching. Standardization includes defining fields to be reformatted (or parsed), normalized, or phonetically encoded. For matching, you must also define the data string to be passed to the match engine. The rules you define for standardization and matching are dependent on the standardization and match engine in use. *Understanding the Oracle Java CAPS Match Engine* describe the rules for the Oracle Java CAPS Match Engine.

You can also configure portions of the match process in the Threshold file, described below, which defines certain match parameters that control weight thresholds, how assumed matches are processed, how potential duplicates are processed, and the query to use for matching.

Threshold

In the Threshold file, you configure the *Manager Service* and define properties of the match process. You specify the match and duplicate thresholds in this file, and define certain system parameters, such as the update mode, how to process records above the match threshold, how to manage same system matches, and whether merged records can be updated. This file also specifies which of the queries defined in the Query Builder to use for matching queries.

This file also configures the EUIDs assigned by the master index application. You can specify an EUID length, whether a checksum value is used for additional verification, and a "chunk size". Specifying a chunk size allows the EUID generator to obtain a block of EUIDs from the sbyn_seq_table database table so it does not need to query the table each time it generates a new EUID.

Best Record

In the Best Record file, you define formulas that determine which data in an enterprise record should be considered the most reliable and how updates to the single best record (SBR) will be handled. The survivor calculator uses these formulas to decide what data from each system record to include in each object's SBR. The SBR is the portion of the enterprise record that represents the data that is considered to be the most accurate and current for an entity.

The SBR is defined by a mapping of fields from external system records. Since there might be many external systems, you can optionally specify a strategy to select the value for an SBR field from the list of external values. You can also specify any additional fields that might be required by the selection strategy to determine which external system contains the best data, such as the object's update date and time.

You can create Java classes that define special processing to perform against a record when the record is created, updated, merged, or unmerged. These classes must be created in the Custom Plug-ins module and can be specified for each transaction type in the Best Record file.

Field Validation

By default, the Field Validation file defines certain validations for the local identifiers assigned by each external system. You can create custom Java classes that define rules for validating field values before they are saved to the master index database. You can then specify the Java classes in the Field Validation file to make them part of the master index application.

Security

This file is not currently used, and is a placeholder to be used in future versions.

Enterprise Data Manager

In the Enterprise Data Manager file, you configure the appearance and processing properties of the Enterprise Data Manager (EDM). In this file, you define each object and field that appears on the EDM, along with the properties of each field, such as the field type and length, field labels, format masks, and so on. You can also define the order in which objects and fields appear on the EDM pages.

This file defines several additional properties of the EDM, including the types of searches available, whether wildcard characters can be used, the criteria for the searches, and the results fields that appear. You can also specify whether an audit log is maintained of each instance data

is accessed through the EDM. For healthcare-based master index applications, such as Oracle Java CAPS Master Patient Index (an application built on the Oracle Java CAPS Master Index platform), this supports the privacy rules mandated by the HIPAA regulation for healthcare.

Finally, the Enterprise Data Manager file defines certain implementation information, such as the application server in use, debugging rules, and security activation.

The files that configure the components of the master index application are created by the wizard and define characteristics of the application, such as how data is processed, queried, and matched, and how it appears on the EDM. These files configure the runtime components of the master index application.

Modifying the Master Index XML Files Directly (Repository)

Make sure that when you modify the configuration files directly, you use the Check Out and Check In commands to maintain version control. Version control is automatic with the Configuration Editor-Repository. If you open and modify a file without first checking the file out, a warning appears when you try to save the file. This warning lets you save and check out the file in one step. Also, be sure to verify that the modifications are valid by verifying the XML syntax. After modifying each file, save the changes.

There are a few restraints on modifying these files. In addition to the general rules listed below, the match or standardization engine you choose might place other requirements on customizations. Be sure to review *Understanding the Oracle Java CAPS Match Engine* before modifying the Match Field file.

Keep the following guidelines in mind when modifying the XML files directly.

- All fields specified in any of the configuration files must be included in the Object Definition file.
- If you add fields to the object structure, make sure you add them to the survivor calculator in the Best Record file.
- If you define additional fields for normalization, parsing, or phonetic encoding, make sure to add the normalized, parsed, and phonetic fields to the Object Definition file and, optionally, the blocking query.
- After modifying any of the configuration files, you must regenerate the master index application and redeploy the project. You must also refresh any client projects that reference the master index server project.

Using the Master Index Configuration Editor-Repository

The Configuration Editor has built in validations to ensure that integrity is maintained between the configuration files. For example, it does not allow you to define a field for normalization if that field is not already defined in the object structure. While you can use the Configuration Editor to modify most of the configurable components, some components can only be modified using the XML editor. Following is a summary of which features can be configured using the Configuration Editor and which need to be modified using an XML editor.

Object Definition File

You can modify most elements of the Object Definition file using the Configuration Editor. The following can only be modified using the XML editor:

- Database type
- Date format
- Maximum field value
- Minimum field value

It is not recommended you change the database type, but if you modify the database type or date format elements, you need to regenerate the application to create the updated database scripts. This does not recreate the Systems or Code Lists scripts; that needs to be done manually.

Candidate Select File

You can modify all elements in the Candidate Select file using the Configuration Editor. If you create a query to use in the EDM or to use for the matching query, you need to add the query to the appropriate file manually (the Threshold file or the Enterprise Data Manager file).

Threshold File

Most elements in the Threshold file cannot be modified using the Configuration Editor. You can only modify the duplicate and match thresholds from the Configuration Editor.

Match Field File

You can use the Configuration Editor to modify all commonly modified elements in the Match Field file, including defining standardization structures, normalization structures, and phonetic encoding. If you create custom classes to implement a block picker, pass controller, match engine, or standardization engine, you need to specify the implementation classes in this file using the XML editor.

Best Record File

The Configuration Editor does not modify the Best Record file. If you make any changes to the object structure (either through the Configuration Editor or XML editor) review this file to verify that all fields or objects are included in the survivor strategy and that the field and object names are correct.

Field Validation File

The Configuration Editor does not modify the Best Record file. If you create a custom field validation class, you need to specify the implementation class in this file using the XML editor.

Enterprise Data Manager File

Most elements in the Enterprise Data Manager file are not modified using the Configuration Editor. You can add and delete fields that appear on the EDM and modify the display name and the value and input masks. All other field properties can only be modified using the XML editor.

Field integrity is maintained when you delete a field using the Configuration Editor. The field is automatically deleted from the EDM object structure and from any EDM page definitions that include the field, such as a search page or report.

Match Configuration File (matchConfigFile.cfg)

You can modify all components of the Match Configuration file using the Configuration Editor, including adding and removing comparators. The Configuration Editor does not validate the extra parameters that can be used for certain comparators, so you should verify your changes by reviewing the match configuration file manually.

Maintaining Version Control in the Master Index (Repository) Configuration Files

When modifying the XML files directly, be sure to maintain version control by checking files out before you modify them and then checking them back in when you are finished. The Configuration Editor supports version control for the XML configuration files. You can manually check the master index configuration files in and out of the Repository, or you can let the Configuration Editor perform version control for you when you open and close the editor.

Checking Configuration Files Out With the Configuration Editor-Repository

If you access the Configuration Editor-Repository when all of the configuration files are already checked out, the Configuration Editor opens immediately. If any of the configuration files are checked in, a dialog box appears that allows you to choose whether to check out and open the files in edit mode or to open the files in view-only mode without checking them out.

Checking Configuration Files In With the Configuration Editor-Repository

After you modify properties in the Configuration Editor, click Save in the Configuration Editor toolbar to save the configuration files to the work space. This does not check the files back in to the Repository. To check the files in, you need to close the editor.

▼ To Check Files In

1. **When you are done making changes, click the Close icon in the upper right corner of the Configuration Editor.**
2. **If there are any unsaved changes, a confirmation dialog box appears. Click Yes to save the changes.**

   The Check In dialog box appears.
3. **Do one of the following:**

   - **To close the editor without checking in the files, click Cancel.**
   - **To check in the files and close the editor, enter a check-in comment and then click Check In.**

     The dialog box and editor close and the files are checked in.

Note – If you close the Configuration Editor without making any changes, a dialog box gives you the option to undo the checkout of the configuration files instead of checking them back in at a new revision level.

Working With the XML Editor

Oracle Java CAPS Master Index supports the version control functionality provided by Java CAPS. You can check files in and out, retrieve older versions to a workspace, view a version history, and so on. In addition, Oracle Java CAPS Master Index supports recursive check-ins and check-outs. When you select Recurse project, you can check in or out all components below the selected node or a subset of those components.

Saving a Configuration File to the Repository

Before modifying a file, be sure to check the file out of the Repository. You can perform this step before or after opening the file. When you are done with your modifications, save the file to the Repository.

▼ To Save a Configuration File to the Repository

1. **With the file open in the XML or text editor, right-click in the XML editor to display the XML editor context menu.**
2. **On the context menu, click Save.**
3. **For XML files, validate the file (described in the following procedure) and check it back in to the Repository.**

Note – If you did not check the file out before making changes and attempting to save it, a warning dialog box appears. Click `Yes` on this dialog box to automatically check out the file, save the changes, and check it back in.

Validating XML Files

Oracle Java CAPS Master Index includes one XML schema definition (XSD) file for each configuration file. Before saving changes to a file, be sure to validate it against the XSD file to make sure no dependencies have been broken during modification.

▼ To Validate XML Syntax

1. **After you save any changes to a configuration file to the Repository, keep the file open and right-click in the text of the file.**
2. **On the context menu that appears, click Check XML.**
3. **A message appears indicating the status of the validation and, if there were errors, includes a list of errors.**
4. **Fix any errors found in the file and revalidate.**

▼ To Validate Against the Schema

1. **After you save any changes to a configuration file to the Repository, right-click that file in the Projects window.**
2. **On the context menu that appears, click Validate.**
3. **A message appears indicating the status of the validation and, if there were errors, includes a list of errors.**
4. **Fix any errors found in the file and revalidate.**

Copying, Cutting, and Pasting Files

You can use standard cut, copy, and paste commands to copy or move files between projects. Oracle Java CAPS Master Index follows the standard Java CAPS functionality, with the exception that you can only copy or move a component from one project into the same node of another project. For example, you can only paste a copied configuration file into the Configuration node of another project. In addition, you cannot cut components that are essential to a project, such as the configuration files, match and standardization files, and so on.

Configuring the Master Index Object Structure (Repository)

After you create the master index framework and create the configuration files, you can modify the object structure that you defined. The properties for the objects and fields you will store in the master index database are defined in the Object Definition file. For more information about this file and the configurable options, see "Master Index Object Definition Configuration (Repository)" in *Understanding Oracle Java CAPS Master Index Configuration Options (Repository)*.

The object tree in the Configuration Editor corresponds to the Object Definition file in the master index application. Any changes you make to the object structure in the Configuration Editor are reflected in the Object Definition file, and EDM field property changes are also reflected in the Enterprise Data Manager file. If you modify the XML file directly, you need to update the Enterprise Data Manager file manually.

Changing the object structure might also require that you make corresponding changes to the other configuration files. For example, if you add a new field to the Object Definition file that you want to include in queries and matching, you need to make the corresponding changes to the Candidate Select file, the Match Field file, and the Best Record file. Only changes made before generating the project take effect for the new application.

You can configure the object structure either through the Configuration Editor-Repository or by modifying the XML file directly. Both methods are described here. Perform any of the following actions to configure the object structure.

Adding an Object to the Master Index Object Structure (Repository)

You can add new objects to the object structure as needed. Note that the object structure can contain only one parent object but multiple child objects. If you use the Configuration Editor, you can either create an undefined object or create an object from a predefined template. A predefined object includes a set of predefined fields with default configurations.

Note – Due to database naming constraints, the length of the name of the parent object plus the length of any child object names must be 21 characters or less.

▼ To Add an Undefined Object (Configuration Editor)

1. **In the Projects window, right-click the master index application you want to modify, and then click Open.**
2. **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

   The Configuration Editor appears.
3. **Select the parent object, and then click Add Sub Object Node on the Configuration Editor toolbar.**

   The object structure expands, and a new child object appears at the bottom of the object structure.
4. **Type a new name for the object, and then press Enter.**
5. **Define the fields for the new object, as described in "Adding a Field to the Master Index Object Structure (Repository)" on page 24.**
6. **On the Configuration Editor toolbar, click Save.**

▼ To Add a Predefined Object (Configuration Editor)

1. **In the Projects window, right-click the master index application you want to modify, and then click Open.**
2. **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

   The Configuration Editor appears.

3. **Select the parent object, click Templates in the toolbar, and then select the template you want to use.**

   The new child object and any defined fields appear in the object tree.

4. **To change the name of the new object, double-click the object name, type the new name, and then press Enter.**

5. **Do any of the following:**

   - **Add fields to the new object, as described in "Adding a Field to the Master Index Object Structure (Repository)" on page 24.**
   - **Delete fields from the new objects, as described in "Deleting a Field from the Master Index Object Structure (Repository)" on page 25.**
   - **Modify the properties of the fields in the object, as described in "Modifying Master Field Properties (Repository)" on page 27.**

6. **On the Configuration Editor toolbar, click Save.**

▼ To Add an Undefined Object (XML editor)

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Object Definition file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the location where you want to create the new object (after the `database` element but before `relationships`).**

3. **Create a `nodes` element.**

4. **Create and name a `tag` element within the new `nodes` element (the value of the `tag` element is the name of the object you are defining).**

   Make sure the new `nodes` element does not fall within any existing `nodes` elements. For example:

```
<nodes>
   <tag>Person</tag>
   ...
</nodes>
<nodes>
   <tag>Address</tag>
</nodes>
```

5. **Define the fields for the new object, as described in "Adding a Field to the Master Index Object Structure (Repository)" on page 24.**

6 Define the relationship of the new object to the existing objects, as described in "Defining Relationships Between Master Index Objects (Repository)" on page 29.

7 Save and close the file.

Modifying an Object's Name In the Master Index Object Definition (Repository)

You can modify the name of an object in the Object Definition file, but you must also make the corresponding changes to the remaining configuration files.

▼ To Modify an Object's Name (Configuration Editor)

1 In the Projects window, right-click the master index application you want to modify, and then click Open.

2 If the Configuration Editor dialog box appears, click Edit to check out the listed files.

The Configuration Editor appears.

3 Click the plus sign next to the parent object to expand the object structure.

4 Click twice on the name of the object you want to change.

5 Type in the new name, and then press Enter.

6 On the Configuration Editor toolbar, click Save.

▼ To Modify an Object's Name (XML Editor)

1 In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Object Definition file.

The file opens in the NetBeans XML editor.

2 Scroll to the `tag` element defining the object you want to modify.

3 Change the value of the `tag` element.

4 Modify the object name in the relationships definition, as described in "Defining Relationships Between Master Index Objects (Repository)" on page 29.

5 Save and close the file.

Deleting an Object From the Master Index Object Structure (Repository)

If you define an object in error, you can remove the object from the Object Definition file. If you modify the XML file directly, you must also remove the relationship definition for the object. Remember to make the corresponding changes to the remaining configuration files.

▼ To Delete an Object (Configuration Editor)

1. **In the Projects window, right-click the master index application you want to modify, and then click Open.**
2. **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

   The Configuration Editor appears.
3. **Expand the object structure by clicking the plus sign by the parent object.**
4. **Do any of the following:**
   - **Right-click in the object tree panel, and then click Delete.**
   - **Press the Delete key.**
   - **In the Configuration Editor toolbar, click Delete.**
5. **On the confirmation dialog box, click Yes.**

   The object and any fields associated with that object are deleted. If you remove the parent object, all child objects are also removed.
6. **On the Configuration Editor toolbar, click Save.**

▼ To Delete an Object (XML Editor)

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Object Definition file.**

   The file opens in the NetBeans XML editor.
2. **Scroll to the `nodes` element containing the object you want to delete.**

3 **Delete all text between and including the `nodes` tags that contain the object tag.**

For example, to delete the Address object below, delete the boldface text.

```
<nodes>
   <tag>Person</tag>
</nodes>
<nodes>
   <tag>Address</tag>
</nodes>
```

4 **Remove the object name from the relationship list, as described in "Defining Relationships Between Master Index Objects (Repository)" on page 29.**

5 **Save and close the file.**

Adding a Field to the Master Index Object Structure (Repository)

Once you define an object in the Object Definition file, you can add new fields to the object and configure the properties for those fields. Be sure to add the field to any relevant structures in the remaining configuration files. For information about field naming restrictions, see "Master Index Field Name Restrictions (Repository)" on page 30.

▼ To Add a Field (Configuration Editor)

1 **In the Projects window, right-click the master index application you want to modify, and then click Open.**

2 **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

The Configuration Editor appears.

3 **Expand the object structure until you see the object to which you want to add a field.**

4 **In the object tree panel, do one of the following:**

- **To add the field to the end of the object's field list, select the name of the object to which you want to add a new field and then click Add Field in the toolbar.**
- **To add the field immediately following an existing field, select the field after which you want to add the new field and then click Add Field in the toolbar.**

  The tree expands and a new field is inserted.

5 **To change the field name, double-click the new field, type the new name, and then press Enter.**

6 **Continue to "Modifying Master Field Properties (Repository)" on page 27.**

▼ To Add a Field (XML Editor)

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Object Definition file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `tag` element defining the object to which you want to add a field.**

3 **Under the `tag` element, create a new `fields` element.**

For example:

```
<nodes>
   <tag>Address</tag>
   <fields>
   </fields>
<nodes>
```

4 **Specify the field properties described in "Master Index Field Property Elements (Repository)" on page 33 within the new `fields` tags.**

For example:

```
<fields>
   <field-name>AddressType</field-name>
   <field-type>string</field-type>
   <size>8</size>
   <updateable>true</updateable>
   <required>true</required>
   <code-module>ADDRTYPE</code-module>
   <pattern/>
   <key-type>true</key-type>
</fields>
```

5 **Save and close the file.**

Deleting a Field from the Master Index Object Structure (Repository)

If a field is defined for an object but does not belong to that object, you can delete the field from the object structure. Make the corresponding changes to the remaining configuration files.

▼ To Delete a Field (Configuration Editor)

1 **In the Projects window, right-click the master index application you want to modify, and then click Open.**

2. **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**
   The Configuration Editor appears.
3. **Expand the object structure until the field you want to delete is visible.**
4. **Select the field and do any of the following:**
   - **Right-click in the object tree panel, and then click Delete.**
   - **Press the Delete key.**
   - **In the Configuration Editor toolbar, click Delete.**
5. **On the confirmation dialog, click Yes.**
   The field is removed from the object tree.
6. **On the Configuration Editor toolbar, click Save.**

▼ To Delete a Field (XML Editor)

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Object Definition file.**
   The file opens in the NetBeans XML editor.
2. **Scroll to the `tag` element defining the object from which you want to delete a field.**
3. **Scroll to the fields element containing the field to delete, and then delete all text between and including the `fields` tags defining that field.**
   For example, to delete the AddressLine1 field below, delete all text in the sample.

```
<fields>
   <field-name>AddressLine1</field-name>
   <field-type>string</field-type>
   <size>5</size>
   <updateable>true</updateable>
   <required>false</required>
   <code-module/>
   <pattern/>
   <key-type>false</key-type>
</fields>
```

4. **Save and close the file.**

Modifying Master Field Properties (Repository)

Every field in the object structure has a set of properties that must be configured before deploying the master index application. When a field is created, a set of default properties are defined for that field. You can modify the property configuration for each field to suit your data processing, storage, and display requirements. Field properties include general field attributes, such as the name, length, and data type, and EDM field properties, such as the display name for the field and display formatting.

Note – On the Configuration Editor, the fields in the EDM section cannot be configured if the field is not defined to appear on the EDM (that is, it does not appear in the Enterprise Data Manager file).

▼ To Modify Field Properties (Configuration Editor)

1. **In the Projects window, right-click the master index application you want to modify, and then click Open.**
2. **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

   The Configuration Editor appears.
3. **Expand the object structure until the field you want to configure is visible.**
4. **In the object structure, select the field to configure.**

   The General Properties page appears.
5. **On the Properties page in the right side of the window, modify the value of any of the properties listed in "Master Index Configuration Editor Field Properties (Repository)" on page 30.**

   Note – After you modify a property value, press Enter to apply the change.

   ---
6. **To view or modify matching properties, click the Matching tab and configure the properties as described in "To Configure the Comparison Functions (Configuration Editor)" on page 118.**

   This tab is only visible for fields that have a value in the Match Type field.
7. **On the Configuration Editor toolbar, click Save.**

▼ To Modify Field Properties (XML Editor)

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Object Definition file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `tag` element defining the object to modify, and then to the fields element containing the field to modify.**

3. **Modify the value of any of the elements described in "Master Index Field Property Elements (Repository)" on page 33.**

   Note – If you modify the name of a field, make the corresponding changes to the remaining configuration files. Some property elements might not exist for a field; add any necessary elements to the field definition to configure the field.

   ---

4. **Save and close the file.**

Example 1 Field Properties in the Object Definition file

The following example defines an address type field that is required in order to enter a record, and that uniquely identifies each address object in a record. It also defines a list, named ADDRTYPE, from which EDM users can select a value to enter into the field.

```
<field-name>AddressType</field-name>
<field-type>string</field-type>
<size>8</size>
<updateable>true</updateable>
<required>true</required>
<pattern></pattern>
<code-module>ADDRTYPE</code-module>
<key-type>true</key-type>
```

The following example defines an employee ID field where the must be equal to or greater than 100001 and less than or equal to 199999. Only the characters 0–9 are allowed.

```
<field-name>EmployeeID</field-name>
<field-type>string</field-type>
<size>6</size>
<updateable>true</updateable>
<required>true</required>
<minimum-value>100001</minimum-value>
<maximum-value>199999</maximum-value>
<pattern>[0-9]{6}</pattern>
<key-type>false</key-type>
```

Defining Relationships Between Master Index Objects (Repository)

Once all objects are customized, you must define relationships between those objects if you are modifying the XML file directly. If you are using the Configuration Editor, the relationships are automatically defined in the object tree.

▼ To Define Object Relationships (XML Editor)

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Object Definition file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `relationships` element near the end of the file.**

3. **To specify a new parent object, modify the value of the `name` element.**

   For example:

   ```
   <name>Individual</name>
   ```

   Note – This is not recommended. Changing the parent name requires changing all instances of the name in all configuration files. To change the parent object name, use the Configuration Editor, which automatically propagates the change.

4. **To change the name of a child object, modify the value of the appropriate `children` element.**

5. **To add a child object, create and name a new `children` element.**

6. **To delete a child object, delete all text between and including the appropriate `children` element.**

7. **Save and close the file.**

   Note – You can only specify one `name` element. The values you specify for the `name` and `children` elements must match an object name specified in the `nodes` elements earlier in the file.

Master Index Field Properties and Name Restrictions (Repository)

Once you create the object structure of the master index application, you can customize several properties for each field, such as whether the field is required, will include a drop-down menu on the EDM, will be used in a blocking query, and so on. There are also some restrictions for how fields can be named and how the properties are defined on the Configuration Editor and in the configuration files.

The following topics provide information about the naming restrictions and about the field properties of the wizard.

Master Index Field Name Restrictions (Repository)

When you name the fields in your object structure, be sure to keep the following guidelines in mind to avoid errors when compiling or running the master index application.

- Oracle Java CAPS Master Index automatically creates a field for each object named *object*Id, where *object* is the name of an object or sub-object. You cannot create fields with those names. For example, you cannot create a field named "AddressId" if there is an Address object in the object structure.
- If you enter a field name longer than 20 characters, a warning dialog box appears. While most databases can handle names up to at least 30 characters, Oracle Java CAPS Master Index appends text to the end of fields defined for phonetic encoding or standardization. For fields that will be parsed, normalized, or phonetically encoded, make sure the name of the original field does not exceed 20 characters. Any other field can have a name up to 30 characters long. For information about the names of the fields automatically created by the wizard, see "Master Index Match Types and Field Names (Repository)" in *Understanding Oracle Java CAPS Master Index Processing (Repository)*.
- Do not use characters or names restricted by the database platform (Oracle or SQL Server), Java, or XML in field names.

Master Index Configuration Editor Field Properties (Repository)

The following table lists and describes the properties you can define on the Field Properties page of the Configuration Editor.

Property	Description
Name	A name for the field. See "Master Index Field Name Restrictions (Repository)" on page 30 for information on valid field names.
Data Type	The data type for each field. Possible data types are: ■ **string** - Fields of this type contain a string of characters. ■ **date** - Fields of this type contain a date value. ■ **float** - Fields of this type contain a floating point integer. ■ **int** - Fields of this type contain an integer. ■ **char** - Fields of this type contain a single character. ■ **boolean** - Fields of this type can contain either **`true`** or **`false`**.
Match Type	The type of matching to be performed against the field, if the field is to be used for match weight generation. You must define at least one field for matching or no weights will be generated. **Note –** The match types you specify here define the structure of the Match Field file, including the match string. The match types in the Match Field file might differ from the match types specified here. See "Master Index Match Types and Field Names (Repository)" in *Understanding Oracle Java CAPS Master Index Processing (Repository)* for information about the available options for this field.
Blocking	An indicator of whether the field will be used in the blocking query. Select this option to add the field to the blocking query; deselect this option to omit the field from the blocking query.
Key Type	An indicator of whether the field is used to identify unique objects. For example, a business index might store several addresses for each business. Each address is assigned an address type and each business can only have one address of each type. Select this option if the field is a unique record identifier; otherwise, deselect it. Key type fields should also be required fields unless a combination of fields are specified as key types for an object. **Note –** It is recommended that each child object contain a key type field, but this is not required. If child objects do not contain one or more key type fields, each enterprise object might accumulate a very large number of child objects depending on the survivor strategy used.
Updateable	An indicator of whether the field can be updated from the EDM and external system messages. Select this option if the field can be updated; otherwise deselect it.
Required	An indicator of whether the field is required in order to save an enterprise object to the database. Select this option if the field is required; otherwise, deselect it.
Field Size	The number of characters allowed in each field. This determines the number of characters allowed in the database columns and defines the maximum number of characters that can be entered into each field on the EDM.

Property	Description
Pattern	The required data pattern for the field. For more information about possible values and using Java patterns, see "Patterns" in the class list for `java.util.regex` in the Javadocs provided with the Java EE Platform. You might want to define patterns for date, telephone, or SSN fields. Note that for the EDM, the pattern is further restricted by the value entered for the input mask described later in this table. If no input mask is specified, all regex patterns are supported.
Code Module	The identification code for the drop-down list that appears for this field in the EDM. **Note –** This must match an entry in the *code* column of the sbyn_common_header database table and, by default, an entry for the code you enter is created in the *Code List* database script. You can further customize code lists directly in the Code List script.
User Code	The processing code for the drop-down list that appears on the EDM for the fields defined by the `constraint–by` property, described below. These codes are used for non-unique IDs, such as account numbers, insurance policies, credit cards, and so on. **Note –** This must match an entry in the *code_list* column of the sbyn_user_code database table.
Constrained By	The name of the field that contains the corresponding User Code value (described above) to use to validate the current field. The User Code and Constrained By properties are used in conjunction to define non-unique ID types, such as credit card numbers or account numbers. The first purpose is to define a drop-down list for the field that contains the User Code value. The second purpose is to validate the field that contains the Constrained By value against definitions for the field with the user code value. For example, if you store non-unique IDs such as credit card numbers or insurance policy numbers, you could create a field named **ID Type** with a User Code of CREDCARD (CREDCARD also needs to be defined as a code in the sbyn_user_code table). This gives the **ID Type** field a drop-down list based on the definitions for CREDCARD in the sbyn_user_code table. Definitions would be VISA, MASTERCARD, AMEX, and so on. You could then create a field named **ID** that would be constrained by the formats defined for the **ID Type** field. Any credit card numbers you enter would be validated against the format defined for the type of credit card you selected in **ID Type**.
Display Name	The name of the field as it will appear on the EDM.

Property	Description
Input Mask	A mask used by the GUI to add punctuation to a field. For example, if users enter the date in the format MMDDYYYY, you can add an input mask to display the dates as MM/DD/YYYY. Use the value mask (described below) to strip the punctuation from the value before storing it in the database. To define an input mask, type a character type for each character in the field and place any necessary punctuation between the character types. For example, the input mask for the above date format is DD/DD/DDDD. Note that the value you enter can further restrict the data pattern for the field (this is the Pattern property). The following character types can be used. ■ **D** – indicates a numeric character. ■ **L** – indicates an alphabetic character. ■ **A** – indicates an alphanumeric character.
Value Mask	A mask used by the index to strip any extra characters that were added by the input mask (see above). This mask ensures that data is stored in the database in the correct format. To specify a value mask, type the same value as is entered for the input mask, but type an "x" in place of each punctuation mark. For example, if an SSN field has an input mask of DDD-DD-DDDD, specify a value mask of DDDxDDxDDDD to strip the dashes before storing the SSN. A value mask is not required for date fields.

Master Index Field Property Elements (Repository)

The following table lists and describes the properties you can define for each field inthe Object Definition file.

Element	Description
field-name	The name of the field. Follow the guidelines under "Master Index Field Name Restrictions (Repository)" on page 30 when naming fields.
field-type	The data type of the field. Possible values are: ■ **string** - Fields of this type contain a string of characters. ■ **date** - Fields of this type contain a date value. ■ **float** - Fields of this type contain a floating point integer. ■ **int** - Fields of this type contain an integer. ■ **char** - Fields of this type contain a single character. ■ **boolean** - Fields of this type can contain either "true" or "false".
size	The number of characters allowed in each field. If you modify this element, be sure to modify the length of the corresponding database column accordingly.

Element	Description
updateable	An indicator of whether the field can be updated from the EDM or by back-end messages. Specify **`true`** if the field can be updated, or specify **`false`** if it cannot.
required	An indicator of whether the field is required in order to save an enterprise object in the database. Specify **`true`** if the field is required, or specify **`false`** if it is not.
code-module	The identification code for the menu list that appears for this field in the EDM. This must match a value in the code column of the sbyn_comment_header database table. This element is optional.
maximum-value	The maximum value allowed in the field. To specify a value for a date field, use the format YYYY-MM-DD. This element is optional.
minimum-value	The minimum value allowed in the field. To specify a value for a date field, use the format YYYY-MM-DD. This element is optional.
pattern	The required pattern for the field. For more information about possible values and using Java patterns, see "Patterns" in the class list for `java.util.regex` in the Javadocs provided with Java EE Platform. You might want to define patterns for date, telephone, or SSN fields. Note that for the EDM, the pattern is further restricted by the value entered for the input mask. If no input mask is specified, all regex patterns are supported. This element is optional.
user-code	The processing code for the drop-down list that appears on the MIDM for the fields defined by the `constraint–by` property, described below. These codes are used for non-unique IDs, such as account numbers, insurance policies, credit cards, and so on. **Note –** This must match an entry in the code_list column of the sbyn_user_code database table.
constraint-by	The name of the field that contains the corresponding `user–code` value (described above) to use to validate the current field. The `user–code` and `constraint-by` properties are used in conjunction to define non-unique ID types, such as credit card numbers or account numbers. The first purpose is to define a drop-down list for the field that contains the user code value. The second purpose is to validate the field that contains the constraint value against definitions for the field with the user code value. For example, if you store non-unique IDs such as credit card numbers or insurance policy numbers, you could create a field named **ID Type** with a `user–code` of CREDCARD (CREDCARD also needs to be defined as a code in the sbyn_user_code table). This gives the **ID Type** field a drop-down list based on the definitions for CREDCARD in the sbyn_user_code table. Definitions would be VISA, MASTERCARD, AMEX, and so on. You could then create a field named **ID** that would be constrained by the formats defined for the **ID Type** field. Any credit card numbers you enter would be validated against the format defined for the type of credit card you selected in **ID Type**.

Element	Description
key-type	An indicator of whether the field is used to identify unique objects. Specify **`true`** if the element is a unique object identifier; specify **`false`** if it is not. Key type fields should also be required fields unless a combination of fields are specified as key types for an object. This element is optional. **Note –** Each child object should contain at least one field that is a unique object identifier, but this is not required. If two or more fields are unique identifiers, the combined value of these fields must be unique in a given enterprise record.

Creating a Master Index Basic Query (Repository)

By default, two basic queries are predefined in the Candidate Select file, one phonetic and one exact alphanumeric. If the default queries do not meet your query requirements, you can define new queries for the master index application. You can either use an existing query builder class or create a custom query builder by creating a custom plug-in (for more information, see "Custom Plug-Ins for Master Index Custom Components (Repository)" in *Developing Oracle Java CAPS Master Indexes (Repository)*).

The changes you make on the Query page of the Configuration Editor are reflected in the Candidate Select file. For more information about this file and the configurable query options, see "Candidate Select Configuration (Repository)" in *Understanding Oracle Java CAPS Master Index Configuration Options (Repository)*. If you create a new query to use from the EDM, make sure to add that query to one of the search definitions in the Enterprise Data Manager file. Unless specifically defined for range searching in the Enterprise Data Manager file, basic queries use exact searching. No configuration is required in the Candidate Select file for basic exact or range searching.

You can create basic queries either through the Configuration Editor or by modifying the XML file directly. Both methods are described here.

▼ To Create a Basic Query (Configuration Editor)

1. **In the Projects window, right-click the master index application you want to modify, and then click Open.**
2. **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

   The Configuration Editor appears.
3. **Click the Query tab.**

   The Query page appears.

4 **In the Basic Queries section, click Add.**

The Basic Query Builder dialog box appears.

5 **Enter values and select options for the fields described in "Master Index Query Builder Dialog Box Fields and XML Elements (Repository)" on page 37.**

6 **On the Configuration Editor toolbar, click Save.**

▼ To Create a Basic Query (XML Editor)

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Candidate Select file.**

The file opens in the NetBeans XML editor.

2 **Create a new `query-builder` element in the `QueryBuilderConfig` element.**

Make sure the new element is created outside of any existing `query-builder` elements.

3 **For the new `query-builder` element, define the attributes listed in "Master Index Query Builder Dialog Box Fields and XML Elements (Repository)" on page 37.**

For example:

```
<query-builder name="PHONETIC-SEARCH" class=
  "com.stc.eindex.user.CustomQueryBuilder"
  parser-class=
  "com.stc.eindex.configurator.impl.querybuilder.KeyValueConfiguration"
   standardize="false" phoneticize="true">
</query-builder>
```

4 **To configure the query to use wildcard characters, do the following:**

a. **Add a new `config` element after the opening `query-builder` element.**

For example:

```
<query-builder name="ALPHA-SEARCH" class=
  "com.stc.eindex.querybuilder.BasicQueryBuilder"
  parser-class=
  "com.stc.eindex.configurator.impl.querybuilder.KeyValueConfiguration"
  standardize="true" phoneticize="true">
   <config>
   </config>
</query-builder>
```

5 **In the new `config` element, create an `option` element and then define `key` and `value` attributes for the new element.**

For example:

```
<config>
 <option key="UseWildcard" value="true"/>
</config>
```

Note – For the default basic query, only the `UseWildcard` parameter is supported. If you create a custom basic query builder, you can use these elements to define additional parameters. For more information, see "Master Index Query Builder Dialog Box Fields and XML Elements (Repository)" on page 37.

6 **Save and close the file.**

Master Index Query Builder Dialog Box Fields and XML Elements (Repository)

The following table lists and describes the fields on the Basic Query Builder and Blocking Query Builder dialog boxes on the Configuration Editor, along with the corresponding elements and attributes in the Candidate Select file.

Configuration Editor Field	XML Element/Attribute	Description
Query Builder Name	query-builder/name	A unique ID that identifies the Query Builder and is referenced from the Enterprise Data Manager file when specifying the query to use on a search page. It is also referenced from the Match Field file when specifying the query to use for matching. No spaces are allowed in the name.
Query Builder Class	query-builder/class	The fully qualified name of the query class. Two default Query Builder classes are provided. ■ `com.stc.eindex.querybuilder.BasicQueryBuilder` – Builds dynamic queries using all the available input fields. When configured to use normalized and phonetic data, this query performs phonetic searches; when configured to not use normalized and phonetic data, this query is used for exact alphanumeric searches. ■ `com.stc.eindex.querybuilder.BlockerQueryBuilder` – Builds queries using the criteria defined in the block definitions defined for the query. When a blocking query is performed, the application searches only on the blocks for which they query has complete data.

Configuration Editor Field	XML Element/Attribute	Description
Parser Class	query-builder/ parser-class	The fully qualified name of the class that parses the configuration elements for each query. The default parser class is `com.stc.eindex.configurator.impl.querybuilder.KeyValueConfiguration`.
Standardize	query-builder/ standardize	An indicator of whether any of the query criteria is standardized before being passed to the query. On the Configuration Editor, select this check box if any search fields are standardized. In the XML file, enter **`true`** if any search fields are standardized; otherwise enter **`false`**.
Phoneticize	query-builder/ phoneticize	An indicator of whether any of the query criteria is phonetically encoded before being passed to the query. On the Configuration Editor, select this check box if any search fields are phonetically encoded. In the XML file, enter **`true`** if any search fields are phonetically encoded; otherwise enter **`false`**.
Use Wildcard	option/key option/value	An indicator of whether wildcard characters are allowed when executing this search. This parameter is available for basic queries only. On the Configuration Editor, select this check box if wildcard characters are allowed in any of the search fields. In the XML file, each `option` element configures one parameter for the query and contains a `key` and `value` pair. The `key` attribute names the parameter. For the default basic query, the name is `UseWildCard`. The `value` attribute specifies the value of the corresponding `key` attribute. For the `UseWildCard` parameter, specify **`true`** to allow wildcard characters for that query type; otherwise specify **`false`**. When wildcard characters are enabled, you can enter a percent sign (%) into search criteria to represent multiple unknown characters. **Note –** If you define a custom query, you can use key and value options to define any required parameters.

Creating Master Index Blocking Queries (Repository)

By default, one blocking query is predefined in the Candidate Select file. If the default query does not meet your requirements, you can define new queries for the master index application. You can either use an existing query builder class or create a custom query builder by creating a custom plug-in (for more information, see "Implementing Master Index Custom Plug-ins (Repository)" in *Developing Oracle Java CAPS Master Indexes (Repository)*). Blocking queries contain *block definitions*, which define the groups of fields used for each query pass and how those groups are processed. Each block definition contains one or more set of *query blocks*. Each query block defines the query rules for one of the groups of fields that make up the block definition.

The changes you make on the Query page of the Configuration Editor are reflected in the Candidate Select file. For more information about this file and the configurable query options, see "Candidate Select Configuration (Repository)" in *Understanding Oracle Java CAPS Master Index Configuration Options (Repository)*. If you create a new query to use from the EDM, make sure to add that query to one of the search definitions in the Enterprise Data Manager file.

You can create blocking queries either through the Configuration Editor or by modifying the XML file directly. Both methods are described here.

▼ To Create a Blocking Query (Configuration Editor)

1. **In the Projects window, right-click the master index application you want to modify, and then click Open.**
2. **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

   The Configuration Editor appears.
3. **Click the Query tab.**

   The Query page appears.
4. **In the Blocking Queries section, click Add.**

   The Blocking Query Builder dialog box appears.
5. **In the Query Builder section, enter the fields described in "Master Index Query Builder Dialog Box Fields and XML Elements (Repository)" on page 37.**
6. **Do the following for each query block you want to include in the query:**

   a. **In the Block Definitions section, click Add.**

      The Block Definition dialog box appears.

   b. **In the Block Name field, enter a unique name for the query block.**

   c. **(Optional) In the Hint field, define an Oracle or SQL Server hint for the query block.**

      For SQL Server, you can only use OPTION hints.

   d. **In the Block By section, click Add.**

      The Block Rule dialog box appears, where you can specify a field to include in the query block.

   e. **Enter values for the fields on the dialog box as described in "Master Index Query Block Fields and XML Elements (Repository)" on page 42, and then click OK.**

f. **For each field to include in the query block, repeat the previous two steps.**

g. **When you are done specifying fields for the rule, click OK on the Block Definition dialog box.**

7 **For each query block you want to create for this query, repeat the previous step.**

8 **When you are done defining the query blocks, click OK.**

9 **On the Configuration Editor toolbar, click Save.**

▼ To Create a Blocking Query (XML Editor)

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Candidate Select file.**

The file opens in the NetBeans XML editor.

2 **Create a new `query-builder` element in the `QueryBuilderConfig` element.**

Make sure the new element is created outside of any existing `query-builder` elements.

3 **For the new `query-builder` element, define the attributes listed in "Master Index Query Builder Dialog Box Fields and XML Elements (Repository)" on page 37.**

For example:

```
<query-builder name="PHONETIC-SEARCH" class=
  "com.stc.eindex.user.CustomQueryBuilder"
  parser-class=
  "com.stc.eindex.configurator.impl.querybuilder.KeyValueConfiguration"
   standardize="false" phoneticize="true">
</query-builder>
```

4 **Create a new `config` element between the `query-builder` tags.**

5 **To create a query block, do the following:**

a. **In the new `config` element, create a new `block-definition` element with a `number` attribute and assign a unique ID code to the `number` attribute.**

For example:

```
<query-builder name="BLOCKING-SEARCH" class=
  "com.stc.eindex.querybuilder.BlockingQueryBuilder"
  parser-class=
  "com.stc.eindex.configurator.impl.querybuilder.
KeyValueConfiguration"
standardize="true" phoneticize="true">
   <config>
      <block-definition number="ID1">
      </block-definition>
```

```
   </config>
</query-builder>
```

b. To add a database hint, create and define a new `hint` element in the new `block-definition` element.

The following example illustrates an Oracle hint.

```
<config>
   <block-definition number="ID1">
   <hint>FIRST_ROWS_100</hint>
   </block-definition>
<config>
```

Tip – Hints are especially useful when a blocking query uses only child object fields; the hint can specify to scan the child object table first. For SQL Server, only OPTION hints are supported.

---

c. In the new `block-definition` element, create a `block-rule` element.

d. For each field in the data block, create the elements and attributes listed in "Master Index Query Block Fields and XML Elements (Repository)" on page 42.

The following example illustrates the use of both `range` and `equals` elements, as well as upper and lower bounds.

```
<config>
   <block-definition number="ID1">
      <hint>FIRST_ROWS_100</hint>
      <block-rule>
         <equals>
            <field>Enterprise.SystemSBR.Person.FirstNamePh</field>
            <source>Person.FirstNamePh</source>
         </equals>
         <equals>
            <field>Enterprise.SystemSBR.Person.LastNamePh</field>
            <source>Person.LastNamePh</source>
         </equals>
         <range>
            <field>Enterprise.SystemSBR.Person.DateOfBirth</field>
            <source>Person.DateOfBirth</source>
            <default>
               <lower-bound type="offset">-5</lower-bound>
               <upper-bound type="constant">2009-01-01
               </upper-bound>
            </default>
         </range>
      </block-rule>
   </block-definition>
<config>
```

6 Repeat the previous step for each data block you need to define for the query.

All data blocks for one query must be defined within one `config` element.

7 Save and close the file.

Master Index Query Block Fields and XML Elements (Repository)

The following table lists and describes the fields on the Block Rule dialog box of the Configuration Editor, along with the corresponding elements and attributes in the Candidate Select file. To see how the XML elements are organized, see the sample in "Adding a Query Block to a Master Index Query (Repository)" on page 47. For more information about the structure of a query block, see "Candidate Select Configuration (Repository)" in *Understanding Oracle Java CAPS Master Index Configuration Options (Repository)*.

Configuration Editor Field	XML Element/Attribute	Description
Field	field	The name of the field to be included in the query block. On the Configuration Editor, you can click Browse to select a field or enter the fully qualified field name manually.

Source	source	The name of the source field in the object from which the criteria is obtained. Click Browse to select a field, or enter the qualified field name manually. An asterisk (*) can be used as a wildcard character.
		Tip – When a field in a child object is defined for a blocking query, use the asterisk wildcard character in the ePath to the source field to ensure all instances of the child object in an incoming message are used as search criteria. Each instance is joined by an OR operator. For example, this configuration:
		`<field>Enterprise.SystemSBR.Person.Name.FirstName</field>`
		`<source>Person.Name[*].FirstName</source>`
		would result in a WHERE clause similar to this:
		`WHERE Name.FirstName="Meg" OR Name.FirstName="Maggie"`

Configuration Editor Field	XML Element/Attribute	Description
Operator	*type of search*	An indicator of the type of search to perform on the field. On the Configuration Editor, you can select one of the following values from the drop-down list. In the XML file, you specify one of the following names for the XML element that defines one field in a query block. ■ **equals** - Performs an exact search against either the criteria or the value defined for the "Use Constants" option. ■ **not-equals** - Searches for values that do not equal either the criteria or the value defined for the "Use Constants" option. ■ **greater-than-or-equal** - Performs a search for values that are greater than or equal to either the criteria or the value defined for the "Use Constants" option. ■ **less-than-or-equal** - Performs a search for values that are less than or equal to either the criteria or the value defined for the "Use Constants" option. ■ **range** - Performs a search against a range of either static or user-defined ranges. If you select this option, you must specify upper and lower bounds. **Tip –** If a field is to be used for simple range searching (where the user or incoming message supplies lower and upper limits of the range are supplied) be sure to define that field for range searching in the Enterprise Data Manager file for the searches that use this query. For more complex range searches that use offset values or constants instead of user-supplied limits, do not define the field for range searching in the Enterprise Data Manager file.
Use Constant/Value	constant	On the Configuration Editor, this is an indicator of whether to use a constant value as the search criteria for the field. When this option is selected, you need to enter the constant in the corresponding Value field. In the XML file, enter the value of the constant in the `constant` element.

Configuration Editor Field	XML Element/Attribute	Description
Upper Bound Type	upper-bound/type	For range searching only, the type of value to use for the upper limit of the search range. Select or enter one of the following options. ■ **not defined** - No specific upper limit is defined; a user enters the value when performing the search. Be sure to define this field for range searching in the EDM as well. In the XML file, this is indicated by omitting the `upper-bound` element. ■ **constant** - A specific value is defined to use as the upper limit of the range when no search criteria is entered or when incomplete information is available. ■ **offset** - A value to be added to the user-supplied value to determine the upper limit on the search range. For constant and offset, enter the value in the corresponding Value field.
Lower Bound Type	lower-bound/type	For range searching only, the type of value to use for the lower limit of the search range. Select one of the options listed for Upper Bound Type. If you select offset, the value you specify for the offset will be subtracted from the user-supplied value.
Value (for the upper and lower bounds)	upper-bound lower-bound	For constant and offset range searching, the upper or lower limit of the range. For constants, this is the upper or lower limit; for offsets, this is the value added to or subtracted from the user-supplied value. It can be numeric, date, or string. In the XML file, the `upper-bound` and `lower-bound` elements fall within an element named `default`.

Modifying Master Index Queries (Repository)

By default, two basic queries and one blocking query are predefined in the Candidate Select file and you can create additional queries. Once a query is defined, you can modify it as needed. If you make changes to a query defined for a search in the EDM, you might need to modify the search definition in the Enterprise Data Manager file. For example, if you deselect the Use Wildcard option for a query and the wildcard feature is enabled for that query in the EDM, you need to disable that feature in the Enterprise Data Manager file.

You can customize the queries used in your implementation by performing any of the following actions.

- "Modifying a Master Index Query (Repository)" on page 45
- "Adding a Query Block to a Master Index Query (Repository)" on page 47
- "Modifying a Query Block for a Master Index Query (Repository)" on page 49
- "Deleting a Query Block From a Master Index Query (Repository)" on page 51

Modifying a Master Index Query (Repository)

Once you define a basic query, you can modify the attributes and parameters for that query. If you defined a custom query, you might need to modify certain elements of the query directly in the XML file instead of through the Configuration Editor.

▼ To Modify a Basic Query (Configuration Editor)

1. **In the Projects window, right-click the master index application you want to modify, and then click Open.**
2. **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

   The Configuration Editor appears.
3. **Click the Query tab.**

   The Query page appears.
4. **Do one of the following:**

   - **To modify a blocking query, select the query to modify in the Blocking Queries section, and then click Edit.**

     The Blocking Query Builder dialog box appears.
   - **To modify a basic query, select the query to modify in the Basic Queries section, and then click Edit.**

     The Basic Query Builder dialog box appears.
5. **Modify any of the fields and configuration options described in "Master Index Query Builder Dialog Box Fields and XML Elements (Repository)" on page 37.**
6. **For blocking queries, you can modify query blocks by performing any of the following procedures:**

   - **"Adding a Query Block to a Master Index Query (Repository)" on page 47**
   - **"Modifying a Query Block for a Master Index Query (Repository)" on page 49**
   - **"Deleting a Query Block From a Master Index Query (Repository)" on page 51**
7. **When you are done editing the query, click OK.**
8. **On the Configuration Editor toolbar, click Save.**

▼ To Modify a Query (XML Editor)

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Candidate Select file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `query-builder` element that contains the query you want to modify.**

3. **Modify any of the attributes listed in "Master Index Query Builder Dialog Box Fields and XML Elements (Repository)" on page 37.**

   For example:

   ```
   <query-builder name="ALPHA-SEARCH" class=
     "com.stc.eindex.querybuilder.MyQueryBuilder"
     parser-class=
     "com.stc.eindex.configurator.impl.querybuilder.KeyValueConfiguration"
      standardize="true" phoneticize="false">
   </query-builder>
   ```

4. **To add a new query parameter (for custom query builders only):**

   a. **Add a new `config` element between the `query-builder` tags.**

   b. **In the new `config` element, create an `option` element and then define `key` and `value` attributes for the new element.**

      For example:

      ```
      <config>
       <option key="SearchAlias" value="true"/>
      </config>
      ```

      See "Master Index Query Builder Dialog Box Fields and XML Elements (Repository)" on page 37 for information about the key and value pairs available to the default basic queries.

5. **To modify a parameter, scroll to the `option` element that defines the parameter and then change the value of the `value` attribute.**

6. **To delete a parameter, scroll to the `config` element of that query, and then delete the `option` element containing the parameter to remove.**

   For example, to remove the SearchAlias parameter in the following sample, delete the boldface text.

   ```
   <config>
      <option key="SearchAlias" value="true"/>
    <option key="UseWildCard" value="false"/>
   </config>
   ```

7 **For blocking queries, you can modify query blocks by performing any of the following procedures:**

- **"Adding a Query Block to a Master Index Query (Repository)" on page 47**
- **"Modifying a Query Block for a Master Index Query (Repository)" on page 49**
- **"Deleting a Query Block From a Master Index Query (Repository)" on page 51**

8 **Save and close the file.**

Adding a Query Block to a Master Index Query (Repository)

Some query blocks might be predefined for you based on information you specified in the wizard. You can create additional query blocks for a blocking query.

▼ To Add a Query Block (Configuration Editor)

1 **In the Projects window, right-click the master index application you want to modify, and then click Open.**

2 **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

The Configuration Editor appears.

3 **Click the Query tab.**

The Query page appears.

4 **In the Blocking Queries section, select the query you want to modify and then click Edit.**

The Blocking Query Builder dialog box appears.

5 **In the Block Definitions section, click Add.**

The Block Definition dialog box appears.

6 **In the Block Name field, enter a unique name for the query block.**

7 **(Optional) In the Hint field, define an Oracle or SQL Server hint for the query block. For SQL Server, you can only use OPTION hints.**

8 **In the Block By section, click Add.**

The Block Rule dialog box appears, where you can specify a field to include in the query block.

9 Enter values for the fields on the dialog box as described in "Master Index Query Block Fields and XML Elements (Repository)" on page 42, and then click OK.

10 Repeat the previous two steps for each field to add to the query block.

11 When you are done, click OK on the Block Definition dialog box.

12 On the Blocking Query Builder dialog box, click OK.

13 On the Configuration Editor toolbar, click Save.

▼ To Add a Query Block (XML Editor)

1 In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Candidate Select file.

The file opens in the NetBeans XML editor.

2 Scroll to the `query-builder` element that contains the blocking query to configure.

3 If a `config` element does not exist for the query, create one between the `query-builder` tags.

4 In the `config` element, create a new `block-definition` element with a `number` attribute and assign a unique ID code to the `number` attribute.

For example:

```
   <config>
      <block-definition number="ID1">
      </block-definition>
   </config>
</query-builder>
```

5 To add a hint, create and define a new `hint` element in the new `block-definition` element.

The following example illustrates an Oracle hint.

```
<config>
   <block-definition number="ID1">
   <hint>FIRST_ROWS_100</hint>
   </block-definition>
<config>
```

6 In the new `block-definition` element, create a `block-rule` element.

7 For each field in the data block, define the elements described in "Master Index Query Block Fields and XML Elements (Repository)" on page 42.

For example:

```
<config>
   <block-definition number="ID1">
```

```
      <hint>FIRST_ROWS_100</hint>
      <block-rule>
         <equals>
            <field>Enterprise.SystemSBR.Person.FirstNamePh</field>
            <source>Person.FirstNamePh</source>
         </equals>
         <equals>
            <field>Enterprise.SystemSBR.Person.LastNamePh</field>
            <source>Person.LastNamePh</source>
         </equals>
         <range>
            <field>Enterprise.SystemSBR.Person.DateOfBirth</field>
            <source>Person.DateOfBirth</source>
            <default>
               <lower-bound type="offset">-5</lower-bound>
               <upper-bound type="constant">2006-01-01
               </upper-bound>
            </default>
         </range>
      </block-rule>
   </block-definition>
<config>
```

8 Save and close the file.

Modifying a Query Block for a Master Index Query (Repository)

Once block definitions are created for a blocking query, you can modify those definitions if needed. You can add, modify, and remove block fields. When adding or removing fields from a query block, verify that the query will still include all the needed fields.

▼ To Modify a Query Block (Configuration Editor)

1 In the Projects window, right-click the master index application you want to modify, and then click Open.

2 If the Configuration Editor dialog box appears, click Edit to check out the listed files.

The Configuration Editor appears.

3 Click the Query tab.

The Query page appears.

4 In the Blocking Queries section, select the query you want to modify and then click Edit.

The Blocking Query Builder dialog box appears.

5 **In the Block Definitions section, select the block to modify and then click Edit.**

The Block Definition dialog box appears.

6 **Do any of the following:**

- **To add a new field, click Add, enter the fields described in "Master Index Query Block Fields and XML Elements (Repository)" on page 42, and then click OK.**
- **To edit an existing field, select the field you want to modify, click Edit, modify any of the fields described in "Master Index Query Block Fields and XML Elements (Repository)" on page 42, and then click OK.**
- **To delete an existing field, select the field you want to delete, click Remove, and then click Yes on the dialog box that appears.**

7 **When you are done modifying the query block, click OK.**

8 **On the Blocking Query Builder dialog box, click OK.**

9 **On the Configuration Editor toolbar, click Save.**

▼ To Modify a Query Block (XML Editor)

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Candidate Select file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `block-rule` element containing the query block you want to modify.**

3 **To add a new blocking field, add and populate the elements described in "Master Index Query Block Fields and XML Elements (Repository)" on page 42.**

For example:

```
<block-rule>
  <equals>
   <field>Enterprise.SystemSBR.Person.DOB</field>
   <source>Person.DOB</source>
  </equals>
  ...
```

4 **To edit an existing field, modify any of the elements described in "Master Index Query Block Fields and XML Elements (Repository)" on page 42.**

5. **To delete an existing field, delete all text between and including the search type elements that define the field (such as `equals`, `range`, `not-equals`, and so on).**

   For example, to delete the DOB field defined above, you would delete the text between and including the `equals` elements.

6. **Save and close the file.**

Deleting a Query Block From a Master Index Query (Repository)

Once you create a block definition, you can delete it if needed. When deleting a query block, verify that the query will still include all the needed fields.

▼ To Delete a Query Block (Configuration Editor)

1. **In the Projects window, right-click the master index application you want to modify, and then click Open.**
2. **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

   The Configuration Editor appears.
3. **Click the Query tab.**

   The Query page appears.
4. **In the Blocking Queries section, select the query you want to modify and then click Edit.**

   The Blocking Query Builder dialog box appears.
5. **In the Block Definitions section, select the definition you want to delete.**
6. **Click Remove.**
7. **On the confirmation dialog box, click Yes.**
8. **On the Configuration Editor toolbar, click Save.**

▼ To Delete a Query Block (XML Editor)

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Candidate Select file.**

   The file opens in the NetBeans XML editor.

2 **Scroll to the `query-builder` element that contains the blocking query to modify.**

3 **In that `query-builder` element, delete all text between and including the `block-definition` element defining the data block to remove.**

For example, in the sample below, to delete the SSN block you would delete the boldface text.

```
<config>
   <block-definition number="ID000007">
      <block-rule>
         <equals>
            <field>Enterprise.SystemSBR.Person.SSN</field>
            <source>Person.SSN</source>
         </equals>
      </block-rule>
   </block-definition>
 <block-definition number="ID000008">
 <block-rule>
          <equals>
             <field>Enterprise.SystemSBR.Person.DOB</field>
             <source>Person.DOB</source>
          </equals>
 </block-rule>
 </block-definition>
```

Note – Each blocking query must contain one `config` element, and the `config` element must contain at least one block definition.

4 **Save and close the file.**

Deleting a Master Index Query (Repository)

Once you create a query, you can delete it if needed. If any of the queries you delete are defined as the matching query in the Threshold file or as a search in the Enterprise Data Manager file, modify those files to point to an existing query. If a defined query is no longer used, you can leave the query in the Candidate Select file as it will not affect processing if you do not delete the query.

Note – The changes you make on the Query page of the Configuration Editor are reflected in the Candidate Select file. For more information about this file and the configurable query options, see "Candidate Select Configuration (Repository)" in *Understanding Oracle Java CAPS Master Index Configuration Options (Repository)*.

▼ To Delete a Query (Configuration Editor)

1 **In the Projects window, right-click the master index application you want to modify, and then click Open.**

2 **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

The Configuration Editor appears.

3 **Click the Query tab.**

The Query page appears.

4 **In the Blocking Queries or Basic Queries section, select the query you want to delete.**

5 **Click Remove.**

6 **On the Configuration Editor toolbar, click Save.**

▼ To Delete a Query

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Candidate Select file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `query-builder` element that contains the query you want to remove.**

3 **Delete all text between and including the `query-builder` element defining the query, including any defined query blocks.**

For example, to delete the query named PHONETIC-SEARCH, you would remove all text below.

```
<query-builder name="PHONETIC-SEARCH"
 class= "com.stc.eindex.querybuilder.BasicQueryBuilder"
 parser-class= "com.stc.eindex.configurator.impl.querybuilder.KeyValueConfiguration"
 standardize="true" phoneticize="true">
 <config>
 <option key="UseWildcard" value="false"/>
 </config>
</query-builder>
```

4 **Save and close the file.**

Configuring Master Index Processing Options (Repository)

The Master Controller defines how transactions are processed, and is configured in the Threshold file. You can specify Java classes that insert additional logic into the match process, configure the blocking query to use for matching, define how merged record updates are handled, and specify whether a record's potential duplicates are reevaluated after it is updated. You can customize the Master Controller by performing any of the following actions.

Specifying Master Index Custom Logic Classes (Repository)

The `logic-class` element specifies custom match processing logic for messages coming from external systems. The and `logic-class-gui` element specifies custom match processing logic for the EDM. If no custom plug-ins were created to define the custom logic, leave these elements empty. Custom logic classes can only be specified by modifying the XML file directly.

▼ To Specify Custom Logic for External System Messages

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Threshold file.**

2. **In the `MasterControllerConfig` element, change the value of the `logic-class` element to the name of the custom plug-in that contains the back-end logic.**

   For example:

   ```
   <logic-class>com.stc.eindex.user.CustomProcessing</logic-class>
   ```

3. **Save and close the file.**

▼ To Specify Custom Logic for the Enterprise Data Manager

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Threshold file.**

2. **In the `MasterControllerConfig` element, change the value of the `logic-class-gui` element to the name of the custom plug-in that contains the EDM logic.**

   For example:

   ```
   <logic-class-gui>com.stc.eindex.user.CustomEDM</logic-class-gui>
   ```

3. **Save and close the file.**

Specifying the Master Index Update Mode (Repository)

The `update-mode` element specifies whether potential duplicates are reevaluated for a record each time the record is updated. If you specify that potential duplicates are reevaluated, the reevaluation only occurs when updates are made to fields involved in blocking and matching. The update mode can only be specified by modifying the XML file directly.

▼ To Specify Potential Duplicates be Reevaluated at Each Update

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Threshold file.**
2. **In the `MasterControllerConfig` element, change the value of the `update-mode` element to `Pessimistic`.**

   For example:

   ```
   <update-mode>Pessimistic</update-mode>
   ```
3. **Save and close the file.**

▼ To Specify Potential Duplicates not be Reevaluated at Each Update

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Threshold file.**
2. **In the `MasterControllerConfig` element, change the value of the `update-mode` element to `Optimistic`.**

   For example:

   ```
   <update-mode>Optimistic</update-mode>
   ```
3. **Save and close the file.**

Configuring Master Index Merged Record Updates (Repository)

The `merged-record-update` element allows you to define whether updates can be made to records with a status of Merged.

▼ To Allow Merged Record Updates

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Threshold file.**

2 **In the `MasterControllerConfig` element, change the value of the `merged-record-update` element to `Enabled`.**

For example:

```
<merged-record-update>Enabled</merged-record-update>
```

3 **Save and close the file.**

▼ To Prevent Merged Record Updates

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Threshold file.**

2 **In the `MasterControllerConfig` element, change the value of the `merged-record-update` element to `Disabled`.**

For example:

```
<merged-record-update>Disabled</merged-record-update>
```

3 **Save and close the file.**

Specifying the Master Index Blocking Query for Matching (Repository)

You need to specify the query that will be used by the master index application to retrieve a candidate selection pool of records that closely match an incoming record and that will be used for probabilistic weighting. The name of the blocking query you specify here must match the name of one of the blocking queries defined in the Candidate Select file. You can only specify one blocking query for matching in the master index application.

Note – Modifying the blocking query once your system is in production is not recommended because it can cause issues with data integrity.

▼ To Specify the Blocking Query for Matching

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Threshold file.**

2. **Scroll to the `execute-match` element in the `MasterControllerConfig` element.**

3. **Modify the value of the `query-builder` name to the name of the blocking query you want to use. For example:**

```
<execute-match>
 <query-builder name="MY-BLOCKER">
 <option key="key" value="value"/>
 </query-builder>
</execute-match>
```

Note – Make sure the name you specify exactly matches the name of one of the blocking queries defined in the Candidate Select file.

---

4. **Set the blocking query options, as described in "Setting Master Index Blocking Query Options (Repository)" on page 57.**

5. **Save and close the file.**

Setting Master Index Blocking Query Options (Repository)

Key and value pairs are designed to fine-tune the operation of custom blocking queries. In the default blocking queries defined in the Candidate Select file, key and value pairs are not used. However, if you create a custom blocking query, you can configure the query to use key and value pairs. The key describes the option, and the value specifies the value of the option.

▼ To Set Blocking Query Options

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Threshold file.**

2. **Scroll to the `execute-match` element.**

3. **To specify a key and value pair, modify the values of the `key` and `value` attributes in the `option` element.**

   For example:

```
<option key="wildcard" value="true"/>
```

4. **To delete a key and value pair, remove the `option` element.**

5. **Save and close the file.**

Configuring Matching Parameters (Repository)

The Decision Maker is configured in the Threshold file and defines how to handle certain decision points in the matching process, such as weight thresholds, how to handle multiple records above the match threshold, and whether records originating from the same system can be automatically matched.

You can customize the Decision Maker by performing any of the following actions:

Specifying the Master Index Decision Maker Class (Repository)

If you create your own Decision Maker class, you can specify the new class to be used by changing the value of the `decision-maker-class` element.

▼ To Specify the Decision Maker Class

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Threshold file.**
2. **Scroll to the `DecisionMakerConfig` element.**
3. **Change the value of the `decision-maker-class` element to the path and name of the new Decision Maker class.**

   For example:

   ```
   <decision-maker-class>com.stc.eindex.decision.impl.MyDecisionMaker
   </decision-maker-class>
   ```

4. **Save and close the file.**

Defining How to Handle Multiple Assumed Matches (OneExactMatch) (Repository)

The `OneExactMatch` parameter determines how records above the match threshold are processed. This parameter can only be modified directly in the XML file. For more information, see "Threshold Configuration (Repository)" in *Understanding Oracle Java CAPS Master Index Configuration Options (Repository)* and "Understanding Master Index Operational Processes (Repository)" in *Understanding Oracle Java CAPS Master Index Processing (Repository)*.

▼ To Create Potential Duplicates When Multiple Records Match

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Threshold file.**
2. **Scroll to the `OneExactMatch` element in the `DecisionMakerConfig` element.**
3. **Change the value of the `parameter-value` element to `true`.**

   For example:

```
<parameter>
   <parameter-name>OneExactMatch</parameter-name>
 <parameter-type>java.lang.Boolean</parameter-type>
 <parameter-value>true</parameter-value>
</parameter>
```

4. **Save and close the file.**

▼ To Match the Highest Weighted Records When Multiple Records Match

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Threshold file.**
2. **Scroll to the `OneExactMatch` element in the `DecisionMakerConfig` element.**
3. **Change value of the `parameter-value` element to `false`.**

   For example:

```
<parameter>
 <parameter-name>OneExactMatch</parameter-name>
 <parameter-type>java.lang.Boolean</parameter-type>
 <parameter-value>false</parameter-value>
</parameter>
```

4. **Save and close the file.**

Specifying Whether Same System Matches are Allowed (SameSystemMatch) (Repository)

The `SameSystemMatch` parameter determines whether two records with local IDs from the same system can be merged automatically. If your local systems contain reliable data, and rarely duplicate their own records, set this parameter to **`true`**. This parameter can only be modified directly through the XML file.

▼ To Allow Same System Records to be Automatically Merged

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Threshold file.**
2. **Scroll to the `SameSystemMatch` element in the `DecisionMakerConfig` element.**
3. **Change the value of the `parameter-value` element to `false`.**

   For example:

```
<parameter>
 <parameter-name>SameSystemMatch</parameter-name>
 <parameter-type>java.lang.Boolean</parameter-type>
 <parameter-value>false</parameter-value>
</parameter>
```

4. **Save and close the file.**

▼ To Prevent Same System Records From Being Automatically Merged

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Threshold file.**
2. **Scroll to the `SameSystemMatch` element in the `DecisionMakerConfig` element.**
3. **Change the value of the `parameter-value` element to `true`.**

   For example:

```
<parameter>
 <parameter-name>SameSystemMatch</parameter-name>
 <parameter-type>java.lang.Boolean</parameter-type>
 <parameter-value>true</parameter-value>
</parameter>
```

4. **Save and close the file.**

Specifying the Master Index Duplicate Threshold (Repository)

The duplicate threshold is the lowest matching probability weight at which two records are considered potential duplicates of one another. Any records with lower probability weights are not considered to be possible matches. Any records between the duplicate and match thresholds are flagged as potential duplicates, and must be resolved manually. You can configure the duplicate threshold by using the Configuration Editor or by modifying the XML file directly.

▼ To Specify the Duplicate Threshold (Configuration Editor)

1. **In the Projects window, right-click the master index application you want to modify, and then click Open.**
2. **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

   The Configuration Editor appears.
3. **Click the Matching tab.**
4. **In the Duplicate Threshold field, enter the lowest weight at which two records should be considered a potential match.**

   Note – This value can be any float value lower than the match threshold but higher than the lowest possible matching probability weight.
5. **On the Configuration Editor toolbar, click Save.**

▼ To Specify the Duplicate Threshold (XML Editor)

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Threshold file.**
2. **Scroll to the `DuplicateThreshold` element in the `DecisionMakerConfig` element.**
3. **Change the value of the `parameter-value` element. For example:**

```
<parameter>
   <parameter-name>DuplicateThreshold</parameter-name>
   <parameter-type>java.lang.Float</parameter-type>
   <parameter-value>7.9</parameter-value>
</parameter>
```

Note – This value can be any float value lower than the match threshold but higher than the lowest possible matching probability weight.

4 **Save and close the file.**

Specifying the Master Index Match Threshold (Repository)

The match threshold specifies the matching probability weight at which two records will be automatically merged, depending on the value of the `OneExactMatch` parameter and the number of records at or above the match threshold. You can configure the match threshold by using the Configuration Editor or by modifying the XML file directly.

▼ To Specify the Match Threshold (Configuration Editor)

1 **In the Projects window, right-click the master index application you want to modify, and then click Open.**

2 **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

The Configuration Editor appears.

3 **Click the Matching tab.**

4 **In the Match Threshold field, enter the lowest weight at which two records should be considered a match.**

Note – This value can be any float value higher than the duplicate threshold but lower than the highest possible matching probability weight.

5 **On the Configuration Editor toolbar, click Save.**

▼ To Specify the Match Threshold (XML Editor)

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Threshold file.**

2 **Scroll to the `MatchThreshold` element in the `DecisionMakerConfig` element.**

3 **Change the value of the `parameter-value` element.**

For example:

```
<parameter>
   <parameter-name>MatchThreshold</parameter-name>
   <parameter-type>java.lang.Float</parameter-type>
   <parameter-value>28.5</parameter-value>
</parameter>
```

Note – This value can be any float value higher than the duplicate threshold but lower than the highest possible matching probability weight.

4 **Save and close the file.**

Adding and Deleting Master Index Decision Maker Parameters (Repository)

New parameters cannot be used with the default Decision Maker class and existing parameters should not be deleted. If you create a custom Decision Maker class, you might need to add new parameters or delete existing ones for the new class. You can only perform these tasks by modifying the XML file directly.

▼ To Add a New Decision Maker Parameter

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Threshold file.**

2 **Scroll to the `DecisionMakerConfig` element.**

3 **Create a `parameter` element inside the `parameters` element, but outside any existing `parameter` elements. Define the following elements:**

- `parameter-name` – The name of the parameter.
- `parameter-type` – The type of parameter. Valid values are `java.lang.Long`, `java.lang.Short`, `java.lang.Byte`, `java.lang.String`, `java.lang.Integer`, `java.lang.Boolean`, `java.lang.Double`, or `java.lang.Float`.
- `parameter-value` – The value of the parameter.

  For example:

```
<parameters>
   <parameter>
      <parameter-name>OneExactMatch</parameter-name>
      <parameter-type>java.lang.Boolean</parameter-type>
      <parameter-value>false</parameter-value>
```

```
      </parameter>
      <parameter>
         <parameter-name>MaxDuplicates</parameter-name>
         <parameter-type>java.lang.Boolean</parameter-type>
         <parameter-value>5</parameter-value>
      </parameter>
   </parameters>
```

4 **Save and close the file.**

▼ To Delete a Decision Maker Parameter

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Threshold file.**

2 **Scroll to the `DecisionMakerConfig` element.**

3 **To delete one parameter, remove all text between and including the parameter element.**

For example, to delete the `ExtensiveSearch` parameter in the following sample, you would delete the boldface text.

```
<parameters>
   <parameter>
      <parameter-name>OneExactMatch</parameter-name>
      <parameter-type>java.lang.Boolean</parameter-type>
      <parameter-value>false</parameter-value>
   </parameter>
   <parameter>
      <parameter-name>ExtensiveSearch</parameter-name>
      <parameter-type>java.lang.Boolean</parameter-type>
      <parameter-value>true</parameter-value>
   </parameter>
</parameters>
```

4 **To remove all parameters, remove all text between and including the starting and ending `parameters` element.**

5 **Save and close the file.**

Configuring Master Index EUIDs (Repository)

The EUID Generator controls how the enterprise-wide unique identifiers (EUIDs) are created by the master index application, including the length of the ID and the checksum value. The EUID generator can only be configured by modifying the Threshold file directly.

You can customize the EUID Generator by performing any of the following actions:

- "Specifying the Master Index EUID Generator Class (Repository)" on page 65
- "Specifying the Master Index EUID Length (Repository)" on page 65

Specifying the Master Index EUID Generator Class (Repository)

The default EUID generator creates sequential EUIDs based on the value specified for the EUID sequence in the sbyn_seq_table database table. If you create a new EUID generator class to process EUIDs differently, you must specify the name of the new class in the `euid-generator-class` element of the Threshold file.

▼ To Specify the EUID Generator Class

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Threshold file.**
2. **Scroll to the `EuidGeneratorConfig` element.**
3. **Change the value of the `euid-generator-class` element to the path and name of the new EUID generator class.**

   For example:

   ```
   <euid-generator-class>com.stc.eindex.idgen.impl.MyEuidGenerator
   </euid-generator-class>
   ```

4. **Save and close the file.**

Specifying the Master Index EUID Length (Repository)

By default, the length of the EUIDs generated by the master index application is 10 characters. You can modify this value if needed. You can also specify a new starting EUID number. This is described in "Creating the Master Index Database (Repository)" in *Developing Oracle Java CAPS Master Indexes (Repository)*.

If you increase the length of the EUIDs to longer than 20 characters or if the EUID length plus the checksum length is longer than 20 characters, you need to increase the length of the EUID columns in the script that creates the database tables.

▼ To Specify the EUID Length

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Threshold file.**
2. **Scroll to the first `parameter` element, IdLength, in the `EuidGeneratorConfig` element.**
3. **Change the value of the `parameter-value` element to the desired length of the EUID.**

   For example:

```
<parameter>
   <parameter-name>IdLength</parameter-name>
   <parameter-type>java.lang.Integer</parameter-type>
   <parameter-value>12</parameter-value>
</parameter>
```

4. **Save and close the file.**

Specifying a Master Index Checksum Length (Repository)

A checksum value is a number appended to the end of each EUID that allows systems to validate EUIDs to ensure accurate identification of records throughout the network. For detailed information about checksum values and how they work, see "Manager Service Components (Repository)" in *Understanding Oracle Java CAPS Master Index Configuration Options (Repository)*.

If you specify **0** (zero), checksum values are not used for validation. If the EUID length plus the checksum length is greater than 20, you need to increase the length of the EUID columns in the database creation scripts.

▼ To Specify a Checksum Length

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Threshold file.**
2. **Scroll to the second `parameter` element, ChecksumLength, in the `EuidGeneratorConfig` element.**
3. **Change the value of the `parameter-value` element to the desired length of the checksum value.**

   For example:

```
<parameter>
   <parameter-name>ChecksumLength</parameter-name>
   <parameter-type>java.lang.Integer</parameter-type>
```

```
   <parameter-value>2</parameter-value>
</parameter>
```

4 **Save and close the file.**

Specifying the Master Index Chunk Size (Repository)

You can specify a number of EUIDs to be allocated at one time to the EUID generator. This allows the generator to assign EUIDs to multiple records without needing to query the sbyn_seq_table database table each time, which improves performance. For detailed information about the chunk size and how EUIDs are allocated, see "Manager Service Components (Repository)" in *Understanding Oracle Java CAPS Master Index Configuration Options (Repository)*.

▼ To Specify the Chunk Size

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Threshold file.**

2 **Scroll to the third `parameter` element, ChunkSize, in the `EuidGeneratorConfig` element.**

3 **Change the value of the `parameter-value` element to the desired chunk size.**

For example:

```
<parameter>
    <parameter-name>EuidGeneratorConfig</parameter-name>
    <parameter-type>java.lang.Integer</parameter-type>
    <parameter-value>100</parameter-value>
</parameter>
```

4 **Save and close the file.**

Adding and Deleting Master Index EUID Generator Parameters (Repository)

You cannot add or delete parameters for the default EUID Generator class, but if you create a custom EUID generator class, you can create additional parameters for the new class and delete the existing parameters if they are not used.

▼ To Add EUID Generator Parameters

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Threshold file.**

2 **Scroll to the `EuidGeneratorConfig` element.**

3 **Create a new `parameter` element inside the `parameters` element, but outside any existing `parameter` elements. Add the following elements:**

- `parameter-name` – The name of the parameter.
- `parameter-type` – The type of parameter. Valid values are `java.lang.Long`, `java.lang.Short`, `java.lang.Byte`, `java.lang.String`, `java.lang.Integer`, `java.lang.Boolean`, `java.lang.Double`, or `java.lang.Float`.
- `parameter-value` – The value of the parameter.

  For example:

```
<parameters>
    <parameter>
      <parameter-name>IdLength</parameter-name>
      <parameter-type>java.lang.Integer</parameter-type>
      <parameter-value>10</parameter-value>
    </parameter>
    <parameter>
      <parameter-name>IncrementBy</parameter-name>
      <parameter-type>java.lang.Integer</parameter-type>
      <parameter-value>5</parameter-value>
   </parameter>
</parameters>
```

4 **Save and close the file.**

▼ To Delete EUID Generator Parameters

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Threshold file.**

2 **Scroll to the `EuidGeneratorConfig` element.**

3 **To delete one parameter, do the following:**

a. **Scroll to the `parameter` element you want to delete.**

b. **Remove all text between and including the parameter element.**

   For example, to remove the `StartNumber` parameter in the sample below, delete all boldface text.

```
<parameters>
    <parameter>
       <parameter-name>IdLength</parameter-name>
       <parameter-type>java.lang.Integer</parameter-type>
       <parameter-value>10</parameter-value>
    </parameter>
    <parameter>
```

```
            <parameter-name>StartNumber</parameter-name>
            <parameter-type>java.lang.Integer</parameter-type>
            <parameter-value>1000000001</parameter-value>
        </parameter>
    </parameters>
```

4 **To remove all parameters, remove all text between and including the `parameters` element.**

5 **Save and close the file.**

Defining Master Index Normalization Rules (Repository)

Normalization is a part of the standardization process, and is the process of changing non-standard values to a common, standard value. For example, the first name a person uses might not be their given name, but might be a nickname instead. To ensure that a proper match is made between first names, nicknames are normalized based on a configurable list. For example, the common value for "Liz" and "Elizabeth" would be "Elizabeth".

Normalization is defined in the Match Field file. You can define normalization by either using the Configuration Editor or modifying the XML file directly. The changes you make on the Normalization page of the Configuration Editor are reflected in the normalization structures of the Match Field file. The Configuration Editor provides a simplified way of defining normalization.

Perform any of the following tasks to define normalization:

- "Defining a Master Index Field to be Normalized (Repository)" on page 69
- "Modifying a Master Index Normalization Definition (Repository)" on page 75
- "Deleting a Master Index Normalization Definition (Repository)" on page 77

Defining a Master Index Field to be Normalized (Repository)

When you define a field for normalization, you define which field contains the data that needs to be normalized and which field will contain the normalized data. You can also specify one or more national domains to use for normalization. A sample normalization structure for the XML file appears at the end of these instructions.

▼ To Define a Field to be Normalized (Configuration Editor)

1 **In the Projects window, right-click the master index application you want to modify, and then click Open.**

2. **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

   The Configuration Editor appears.

3. **In the object structure in the left pane, add the field that will contain the normalized value.**

   For more information, see "Adding a Field to the Master Index Object Structure (Repository)" on page 24.

4. **Click the Normalization tab.**

   The Normalization page appears.

5. **Click Add.**

   The Normalized Field dialog box appears.

6. **Enter or select a value for each of the fields described in "Master Index Normalization and Standardization Structure Properties (Repository)" on page 73.**

7. **To specify a national domain for the type of data being standardized, do the following:**

   a. **In the Locale Field Name field, select the field whose value in incoming records will indicate which variant to use.**

   b. **In the Locale Codes section, click Add.**

   c. **On the dialog box that appears, enter values in the fields described in "Master Index Locale Codes Properties (Repository)" on page 75.**

   d. **Click OK.**

      If you selected the multiple domain selector, you can add multiple national domains; otherwise, you can add one default national domain and one field-defined national domain.

8. **On the Normalized Field dialog box, click OK.**

   The new normalization definition appears in the list.

9. **On the Configuration Editor toolbar, click Save.**

▼ To Define a Field to be Normalized (XML Editor)

Before You Begin In the Object Definition file, create the field that will contain the new normalized value. For more information, see "Adding a Field to the Master Index Object Structure (Repository)" on page 24.

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Match Field file.**

   The file opens in the NetBeans XML editor.

2. **In the `structures-to-normalize` element, create and name a new `group` element.**

   Make sure the new element falls within the `structures-to-normalize` element, but outside any existing `group` tags.

3. **In the new `group` element, define the `standardization-type` and `domain-selector` attributes (these are described in "Master Index Normalization and Standardization Structure Properties (Repository)" on page 73.**

4. **If you specified the multiple domain selector for the `domain-selector` attribute, do the following:**

   a. **In the `group` element, create a `locale-field-name` element and a `locale-maps` element (described in "Master Index Normalization and Standardization Structure Properties (Repository)" on page 73 and "Master Index Locale Codes Properties (Repository)" on page 75).**

   b. **For each variant you want to use, define a `locale-codes`, `value`, and `locale` element in the `locale-maps` element (described in "Master Index Locale Codes Properties (Repository)" on page 75).**

5. **To specify the source fields to normalize, do the following:**

   a. **Create a new `unnormalized-source-fields` element in the `group` element.**

   b. **Create a `source-mapping` element in the new `unnormalized-source-fields` element.**

   c. **Define the `unnormalized-source-field-name` and `standardized-object-field-id` elements (these are described in"Master Index Normalization and Standardization Structure Properties (Repository)" on page 73).**

6. **To map the normalized data to destination fields, do the following:**

   a. **Create a new `normalization-targets` element under the `unnormalized-source-fields` element that defines the field to map.**

   b. **Create a `target-mapping` element in the new `normalization-targets` element.**

   c. **Define the `standardized-object-field-id` and `standardized-target-field-name` elements (these are described in "Master Index Normalization and Standardization Structure Properties (Repository)" on page 73).**

7 Save and close the file.

Example 2 First and Last Name Normalization

```
<structures-to-normalize>
         <group standardization-type="PersonName" domain-selector=
          "com.stc.eindex.matching.impl.MultiDomainSelector">
            <locale-field-name>Person.PobCountry</locale-field-name>
            <locale-maps>
              <locale-codes>
                <value>GB</value>
                <locale>UK</locale>
              </locale-codes>
              <locale-codes>
                <value>UNST</value>
                <locale>US</locale>
              </locale-codes>
              <locale-codes>
                <value>Default</value>
                <locale>US</locale>
              </locale-codes>
            </locale-maps>
            <unnormalized-source-fields>
               <source-mapping>
                  <unnormalized-source-field-name>
                   Person.Alias[*].FirstName
                  </unnormalized-source-field-name>
                  <standardized-object-field-id>FirstName
                  </standardized-object-field-id>
               </source-mapping>
               <source-mapping>
                  <unnormalized-source-field-name>
                   Person.Alias[*].LastName
                  </unnormalized-source-field-name>
                  <standardized-object-field-id>LastName
                  </standardized-object-field-id>
               </source-mapping>
            </unnormalized-source-fields>
            <normalization-targets>
               <target-mapping>
                  <standardized-object-field-id>FirstName
                  </standardized-object-field-id>
                  <standardized-target-field-name>
                     Person.Alias[*].StdFirstName
                  </standardized-target-field-name>
               </target-mapping>
               <target-mapping>
                  <standardized-object-field-id>LastName
                  </standardized-object-field-id>
                  <standardized-target-field-name>
                     Person.Alias[*].StdLastName
                  </standardized-target-field-name>
               </target-mapping>
            </normalization-targets>
         </group>
```

Master Index Normalization and Standardization Structure Properties (Repository)

The following table lists and describes the Configuration Editor fields and their corresponding XML elements that define the fields to be normalized or standardized in the master index application.

You can specify one or more national domains for data to be standardized. For a single national domain, you only need to specify the national domain if you need to standardize data that is not from the United States. If you are standardizing data from multiple countries, use the multiple domain selector. This requires that one field in the object structure identify which national domain to use for each field that will be standardized. For example, the value of the Country field in a system record could be used to tell the standardization engine which national domain to use for a particular set of data. If you specified the multiple domain selector in the `domain-selector` element, you must also define the identifying field and then map the values that can be populated into that field to their corresponding national domain.

The following rules apply to the multiple domain selector:

- You can specify a value of "Default" for the identifying field. The corresponding national domain is used if the identifying field is blank, contains the value "Default", or contains a value not defined by any of the `value` elements.
- If a "Default" value is not defined, the system default national domain, United States, is used as the default.

For more information about the fields and elements described in the following table, see *Understanding the Oracle Java CAPS Match Engine*.

Configuration Editor Field	XML File Element or Attribute	Description
Type	standardization-type	The type of standardization to perform on the source fields. This is specific to the type of data being processed.

Configuration Editor Field	XML File Element or Attribute	Description
Domain Selector	domain-selector	The Java class used by the standardization engine to determine the national domain of the data being processed. For the Oracle Java CAPS Match Engine, the following classes can be specified. If no selector is specified, the default is US. The Oracle Java CAPS Match Engine supports Australian, French, United Kingdom, and United States national domains. Possible values for this field are: ■ com.stc.eindex.matching.impl.SingleDomainSelectorAU ■ com.stc.eindex.matching.impl.SingleDomainSelectorFR ■ com.stc.eindex.matching.impl.SingleDomainSelectorUK ■ com.stc.eindex.matching.impl.SingleDomainSelectorUS ■ com.stc.eindex.matching.impl.MultipleDomainSelector
Locale Field Name	locale-field-name	The ePath to an identifying field in the object structure that identifies which of the defined national domains (element `locale-codes`) to use. If no field is specified for the Oracle Java CAPS Match Engine, the standardization engine defaults to the United States, regardless of whether any national domains are defined. This field must be contained in the object that contains the fields defined for normalization in this structure.
Unnormalized Source	unnormalized-source-field-name	The field that contains the data to be normalized. The field is designated by its ePath (for example, Person.FirstName).
Unnormalized Standardization Component	standardized-object-field-id	An identification code that identifies the field to normalize to the standardization engine. This ID is specific to the standardization engine and must correspond to a standardization component defined by that engine.
Normalized Standardization Component	standardized-object-field-id	An identification code that identifies the field that contains the normalized data to the standardization engine. This is specific to the standardization engine in use and must correspond to a standardization component defined by that engine.

Configuration Editor Field	XML File Element or Attribute	Description
Normalized Target	standardized-target-field-name	The field that will store the normalized data. The field is designated by its ePath (for example, Person.Alias[*].StdLastName).

Master Index Locale Codes Properties (Repository)

The following table lists and describes the Configuration Editor fields and XML elements that define a national domain for normalization or standardization. In the XML file, each value and locale pair are defined within a `locale_codes` element. A list of `locale_codes` elements can be defined in the `locale_maps` element.

Configuration Editor Field	XML File Element or Attribute	Description
Value	value	A value that indicates to the standardization engine which national domain to use to standardize the data. When the value is contained in the Locale Field Name field (or the `locale-field-name` element), the standardization engine uses the corresponding Locale field (or locale element) to determine the national domain. To specify a default national domain, enter "Default".
Locale	locale	A code indicating which national domain to use to standardize data when the identifying field value in a transaction matches the corresponding Value field or element. Select one of the following codes. ■ **AU** - Australia ■ **FR** - France ■ **UK** - United Kingdom ■ **US** - United States

Modifying a Master Index Normalization Definition (Repository)

Once you create a normalization definition, you can modify it as needed. Use caution when modifying normalization definitions once a system is in production. This can cause inconsistent match results.

▼ To Modify a Normalization Definition (Configuration Editor)

1. **In the Projects window, right-click the master index application you want to modify, and then click Open.**
2. **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**
   The Configuration Editor appears.
3. **Click the Normalization tab.**
   The Normalization page appears.
4. **In the Normalization Mappings list, click the definition you want to modify.**
5. **Click Edit.**
6. **Do any of the following:**
   - **Modify any of the fields described in "Master Index Normalization and Standardization Structure Properties (Repository)" on page 73.**
   - **To modify a national domain, select the national domain under Locale Codes, and then click Edit. Modify either field on the dialog box that appears.**
   - **To remove a national domain, select the national domain under Locale Codes, and then click Remove. Click Yes on the dialog box that appears.**
   - **Click OK.**
7. **On the Configuration Editor toolbar, click Save.**

▼ To Modify a Normalization Structure (XML Editor)

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Match Field file.**
   The file opens in the NetBeans XML editor.
2. **Scroll to the `structures-to-normalize` element in the `StandardizationConfig` element.**
3. **To modify the normalization type, change the value of the `standardization-type` attribute.**
4. **To change the national domain, change the value of the `domain-selector` element as described in "Master Index Normalization and Standardization Structure Properties (Repository)" on page 73.**

5. **To modify an existing source field, scroll to the `unnormalized-source-fields` element in the appropriate `group` element, and then change the value of any source field elements (these are described in "Master Index Normalization and Standardization Structure Properties (Repository)" on page 73).**

6. **To modify an existing destination field, scroll to the `normalization-targets` element in the appropriate `group` element, and then change the value of any target field elements (these are described in "Master Index Normalization and Standardization Structure Properties (Repository)" on page 73).**

7. **Save and close the file.**

Deleting a Master Index Normalization Definition (Repository)

If a defined normalization structure is not needed, you can delete the normalization structure from the standardization configuration. If no data requires normalization, you can delete all normalization structures. It is not recommend that you delete a normalization definition once a system is in production. This can cause inconsistent match results.

▼ To Delete a Normalization Definition

1. **In the Projects window, right-click the master index application you want to modify, and then click Open.**

2. **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

   The Configuration Editor appears.

3. **In the Configuration Editor toolbar, click the Normalization tab.**

   The Normalization page appears.

4. **In the Normalization Mappings list, click the definition you want to delete.**

5. **Click Remove.**

6. **On the Configuration Editor toolbar, click Save.**

▼ To Delete a Normalization Structure

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Match Field file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `structures-to-normalize` element in the `StandardizationConfig` element.**

3. **Do either of the following:**

   - **To delete an existing normalization structure, delete all text between and including the `group` element that defines the structure.**
   - **To specify that no objects require normalization, delete all text between, but not including, the `structures-to-normalize` element.**

4. **Save and close the file.**

Defining Master Index Standardization Rules (Repository)

If any of the fields against which searching or matching is performed are entered in free-form text format, those fields must be standardized before being sent to the standardization engine. The process of standardization includes reformatting, or parsing, the input data field and then normalizing some of the parsed data to a standard value. For example, street addresses can be parsed into the house number, street name, street type, and so on. The street name and type can then be normalized to their commonly used values. "Ave" might be normalized to "Avenue", "St." to "Street", and so on.

Standardization is defined in the Match Field file. You can define standardization by either using the Configuration Editor or modifying the XML file directly. The changes you make on the Standardization page of the Configuration Editor are reflected in the standardization structures of the Match Field file. The Configuration Editor provides a simplified way of defining standardization.

Perform any of the following tasks to define standardization:

- "Defining Master Index Fields to be Standardized (Repository)" on page 79
- "Modifying a Master Index Standardization Definition (Repository)" on page 83
- "Deleting a Master Index Standardization Definition (Repository)" on page 86

Defining Master Index Fields to be Standardized (Repository)

When you define fields for standardization, you can specify the type of standardization to perform on each field or group of fields, the nationality of the data, and a field that indicates which nationality to use (if you specify more than one). You also specify which fields contain the data that needs to be parsed and normalized, and which fields contain the parsed and normalized data. For each standardization structure, you can specify more than one source field, but they must use the same standardization type. The source fields in one standardization structure are concatenated before being parsed.

A sample standardization structure for the XML file is included at the end of these instructions.

▼ To Define Fields to be Standardized (Configuration Editor)

1. **In the Projects window, right-click the master index application you want to modify, and then click Open.**

2. **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

   The Configuration Editor appears.

3. **In the object structure in the left pane, create the fields that will contain the parsed components of the new field to be standardized.**

   For more information, see "Adding a Field to the Master Index Object Structure (Repository)" on page 24.

4. **Click the Standardization tab.**

   The Standardization page appears.

5. **Click Add.**

   The Standardization Type dialog box appears.

6. **Enter values for the Type, Domain Selector, and Locale Field Name fields (these are described in "Master Index Normalization and Standardization Structure Properties (Repository)" on page 73.**

7. **To define a national domain for the standardization engine to use, do the following:**

   a. **In the Locale Codes section, Click Add.**

   b. **On the Locale Codes dialog box, enter values in the fields described in "Master Index Locale Codes Properties (Repository)" on page 75.**

c. Click OK.

If you selected the multiple domain selector, you can add multiple national domains; otherwise, you can define one default national domain and one defined national domain.

8 Under Source Fields to be Standardized, click Add.

The Select Source Field(s) dialog box appears.

9 In the left panel, select the field that contains the data that needs to be parsed and normalized, and then click the right arrow.

Note – If the data is contained in more than one field, select all fields that contain the data. For example, a street address might be contained in two fields, such as Street Address and Unit. Both fields should be selected for standardization; they will be concatenated during the standardization process.

---

10 If you add a field in error, select the field in the Selected Source Field(s) list, and then click the left arrow.

11 Click OK.

12 For each field in which the parsed and normalized data will be stored, do the following:

a. On the Standardized Fields dialog box, click Add under Target Mappings.

The Target Mapping dialog box appears.

b. In the Select Target field, select the name of a field that will contain standardized data.

c. In the Available Standardization Components list, select the ID associated with the field, and then click Add between the left and right panels.

d. To change the priority of a component in the Selected Standardization Components list, select the component and then click Move Up or Move Down.

e. If you add a component in error, select the component in the Selected Standardization Components list, and then click Remove.

f. Click OK.

Note – For more information about standardization components and the fields to which they pertain, see *Understanding the Oracle Java CAPS Match Engine*.

---

13 **Click OK on the Standardization Type dialog box.**

The new standardization definition appears in the list.

14 **On the Configuration Editor toolbar, click Save.**

▼ To Define Fields to be Standardized (XML Editor)

Before You Begin In the Object Definition file, create the fields that will contain the parsed components of the field to be standardized. For more information, see "Adding a Field to the Master Index Object Structure (Repository)" on page 24.

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Match Field file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `free-form-texts-to-standardize` element in the `StandardizationConfig` element.**

3 **Create a new `group` element in the `free-form-texts-to-standardize` element, and then define the `standardization-type` and `domain-selector` attributes (these are described in "Master Index Normalization and Standardization Structure Properties (Repository)" on page 73).**

Make sure the new element falls within the `free-form-texts-to-standardize` element, but outside any existing `group` tags.

4 **If you specified the multiple domain selector for the `domain-selector` attribute, do the following:**

   a. **In the `group` element, create a `locale-field-name` element and a `locale-maps` element.**

   b. **Define the elements described in "Master Index Locale Codes Properties (Repository)" on page 75).**

5 **To specify the source fields to standardize, do the following:**

   a. **If it does not currently exist, create an `unstandardized-source-fields` element in the appropriate `group` element (each group element can only include one `unstandardized-source-fields` element).**

   b. **For each field standardized by the specified standardization type, create and name a new `unstandardized-source-field-name` element in the new `unstandardized-source-fields` element.**

Note – If more than one source field is defined, the fields are concatenated prior to standardization (with a pipe (|) between them for the Oracle Java CAPS Match Engine). If you want the fields to be processed separately, you need to create two standardization structures. Source fields are designated by their ePaths.

6 **To specify the destination fields for the standardized data, do the following:**

a. **In the `group` element for which destination fields need to be defined, create a `standardization-targets` element after the `unstandardized-source-fields` element.**

b. **In the new element, create a `target-mapping` element for each destination field, and then define the last two elements described in "Master Index Standardization Source and Target Field Elements (Repository)" on page 83.**

7 **Save and close the file.**

Example 3 Address Standardization Structure

```
<group standardization-type="Address" domain-selector=
 "com.stc.eindex.matching.impl.SingleDomainSelectorUS">
  <locale-field-name>Person.Address[*].CountryCode
  </locale-field-name>
  <locale-maps>
     <locale-codes>
         <value>GB</value>
         <locale>UK</locale>
      </locale-codes>
      <locale-codes>
         <value>UNST</value>
         <locale>US</locale>
      </locale-codes>
      <locale-codes>
         <value>AU</value>
         <locale>AU</locale>
      </locale-codes>
      <locale-codes>
         <value>Default</value>
         <locale>AU</locale>
      </locale-codes>
   </locale-maps>
   <unstandardized-source-fields>
      <unstandardized-source-field-name>Person.Address[*].AddressLine1
      </unstandardized-source-field-name>
      <unstandardized-source-field-name>Person.Address[*].AddressLine2
      </unstandardized-source-field-name>
   </unstandardized-source-fields>
   <standardization-targets>
      <target-mapping>
         <standardized-object-field-id>HouseNumber
         </standardized-object-field-id>
         <standardized-target-field-name>Person.Address[*].HouseNumber
```

```
            </standardized-target-field-name>
         </target-mapping>
         <target-mapping>
            <standardized-object-field-id>MatchStreetName
            </standardized-object-field-id>
            <standardized-target-field-name>Person.Address[*].StreetName
            </standardized-target-field-name>
         </target-mapping>
      </standardization-targets>
</group>
```

Master Index Standardization Source and Target Field Elements (Repository)

The following table lists and describes the XML elements that define the source and target fields for standardization. The data from the source fields is standardized, and the standardized values are stored in the target fields.

XML File Element or Attribute	Description
unstandardized-source-field-name	The field or fields that contain the data to be standardized. The field is designated by its ePath (for example, Person.FirstName).
standardized-object-field-id	An code that identifies the standardized component from the source field to store in the target field. This is specific to the standardization engine in use and must correspond to a standardization component defined by that engine. For more information, see *Understanding the Oracle Java CAPS Match Engine*.
standardized-target-field-name	The field that stores the standardized data. You can have multiple target fields, depending on how much of the standardized data you want to store. The fields are designated by their ePaths (for example, Person.Alias[*].StdLastName).

Modifying a Master Index Standardization Definition (Repository)

You can modify an existing standardization definition. Use caution when modifying standardization after a system is in production because it can cause inconsistent matching results.

▼ To Modify a Standardization Definition (Configuration Editor)

1 **In the Projects window, right-click the master index application you want to modify, and then click Open.**

2 **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

The Configuration Editor appears.

3 **Click the Standardization tab.**

The Standardization page appears.

4 **In the Standardization Types list, select the definition you want to modify, and then click Edit.**

The Standardization Type dialog box appears.

5 **Do any of the following:**

- **Modify any of the fields or perform any of the functions described in "Defining Master Index Fields to be Standardized (Repository)" on page 79.**
- **To modify a national domain, select the code under Locale Codes, and then click Edit. Modify either field on the dialog that appears.**
- **To remove a national domain, select the code under Locale Codes, and then click Remove. Click Yes on the dialog box that appears.**
- **To remove a source field, select the field under Source fields to be standardized, and then click Remove. Click Yes on the dialog box that appears.**

  Note – There must be at least one field in this list.

- **To edit a target field, select the field in the Specifying Target Mappings list and then click Edit.**

  Note – You can select new components, move selected components up and down in priority, and remove components.

- **To delete a target field, select the field in the Specifying Target Mappings list, and then click Remove. Click Yes on the dialog box that appears.**

6 **When you are done making changes, click OK on the Standardization Type dialog box.**

7 **On the Configuration Editor toolbar, click Save.**

▼ To Modify a Standardization Definition (XML Editor)

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Match Field file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `structures-to-normalize` element in the `StandardizationConfig` element.**

3. **To modify the standardization type, change the value of the `standardization-type` attribute.**

4. **To change the national domain, change the value of the `domain-selector` element (described in "Master Index Normalization and Standardization Structure Properties (Repository)" on page 73).**

5. **To modify an existing source field, scroll to the appropriate `group` element, and then change the value of the `unstandardized-source-field-name` element to the ePath of the new field.**

6. **To modify an existing destination field, scroll to the `target-mapping` element in the `standardization-targets` section, and then change the value of either target mapping element (these are the last two elements described in "Master Index Standardization Source and Target Field Elements (Repository)" on page 83).**

7. **To remove an existing source field, delete all text between and including the `unstandardized-source-field-name` element that defines the field.**

   Note – If no fields require standardization in a defined standardization structure, delete the entire structure as described in "Deleting a Master Index Standardization Definition (Repository)" on page 86.

   ---

8. **To remove an existing destination field, delete all text between and including the `target-mapping` tags that define the field.**

   Note – Each standardization structure must have at least one destination field defined for standardized data. If a structure does not contain any fields that need to be standardized, you can delete the entire structure, as described in "Deleting a Master Index Standardization Definition (Repository)" on page 86.

   ---

9. **Save and close the file.**

Deleting a Master Index Standardization Definition (Repository)

You can delete an existing standardization definition. It is not recommended that a standardization definition be deleted after a system is in production since this can cause inconsistent matching results.

▼ To Delete a Standardization Definition (Configuration Editor)

1. **In the Projects window, right-click the master index application you want to modify, and then click Open.**
2. **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

   The Configuration Editor appears.
3. **Click the Standardization tab.**

   The Standardization page appears.
4. **In the Standardization Types list, select the definition you want to delete.**
5. **Click Remove.**
6. **Click Yes on the dialog box that appears.**
7. **On the Configuration Editor toolbar, click Save.**

▼ To Delete a Standardization Definition (Configuration Editor)

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Match Field file.**

   The file opens in the NetBeans XML editor.
2. **Scroll to the `free-form-texts-to-standardize` element in the `StandardizationConfig` element.**

3 **Do either of the following:**

- **To delete an existing standardization structure, delete all text between and including the `group` element that defines the structure.**

  Using the example below, to delete the Address object, delete all boldface text.

```
<free-form-texts-to-standardize>
   <group standardization-type="BusinessName" domain-selector=
    "com.stc.eindex.matching.impl.SingleDomainSelectorUS">
      ...
   </group>
   <group standardization-type="Address" domain-selector=
    "com.stc.eindex.matching.impl.SingleDomainSelectorUS">
      ...
   </group>
</free-form-texts-to-standardize>
```

- **To specify that no fields require standardization, delete all text between, but not including, the `free-form-texts-to-standardize` element.**

  This deletes all standardization structures.

4 **Save and close the file.**

Defining Phonetic Encoding for the Master Index (Repository)

Oracle Java CAPS Master Index provides configurable phonetic encoding capabilities. Phonetic encoding is a part of the standardization process, and is the process of changing the value of a data field to its phonetic equivalent. It is used to retrieve records with similar field values from the database for matching. You can specify which fields are phonetically encoded before matching and how they are encoded. There are several different encoders you can use for this purpose. This is most commonly done for first names and street names.

Phonetic encoding is defined in the Match Field file. You can define phonetic encoding by either using the Configuration Editor or modifying the XML file directly. The changes you make on the Phoneticized Field page of the Configuration Editor are reflected in the phonetic encoding structures and the match service section of the Match Field file. The Configuration Editor provides a simplified way of defining phonetic encoding.

Perform any of the following tasks to define phonetic encoding rules:

- "Defining Master Index Fields for Phonetic Encoding (Repository)" on page 88
- "Modifying a Master Index Phonetic Encoding Definition (Repository)" on page 90
- "Deleting a Master Index Phonetic Encoding Definition (Repository)" on page 91
- "Defining a Master Index Phonetic Encoder (Repository)" on page 92
- "Modifying a Master Index Phonetic Encoder (Repository)" on page 94
- "Deleting a Master Index Phonetic Encoder (Repository)" on page 95

Defining Master Index Fields for Phonetic Encoding (Repository)

You can specify the fields you want to be phonetically encoded, the fields that store the encoded values, and the type of phonetic encoder to use for each field, such as NYSIIS or Soundex. A sample phonetic encoding structure for the XML file is included at the end of these instructions.

▼ To Define a Field for Phonetic Encoding (Configuration Editor)

1. **In the Projects window, right-click the master index application you want to modify, and then click Open.**
2. **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

   The Configuration Editor appears.
3. **In the object structure in the left pane, create the field that will contain the phonetic value of the field to be encoded.**

   For more information, see "Adding a Field to the Master Index Object Structure (Repository)" on page 24.
4. **Click the Phoneticized Fields tab.**

   The Phoneticized Field page appears.
5. **In the Phoneticized Fields section, click Add.**

   The Phoneticized Field dialog box appears.
6. **Select values for the fields described in "Master Index Phonetic Encoding Fields and Elements (Repository)" on page 89.**
7. **Click OK.**

   The phonetic encoding definition is added to the Phoneticized Fields list.
8. **On the Configuration Editor toolbar, click Save.**

▼ To Define a Field for Phonetic Encoding (XML Editor)

In the Object Definition file, create the field that will contain the phonetic value. For more information, see "Adding a Field to the Master Index Object Structure (Repository)" on page 24.

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Match Field file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `phoneticize-fields` element in the `PhoneticEncodersConfig` element.**

3. **Create a new `phoneticize-field` element within the `phoneticize-fields` element.**

4. **In the new `phoneticize-field` element, create and define the elements described in "Master Index Phonetic Encoding Fields and Elements (Repository)" on page 89.**

5. **Save and close the file.**

Example 4 Phonetic Encoding Structure

```
<phoneticize-fields>
   <phoneticize-field>
      <unphoneticized-source-field-name>Person.FirstName
      </unphoneticized-source-field-name>
      <phoneticized-target-field-name>Person.FirstNamePhoneticCode
      </phoneticized-target-field-name>
      <encoding-type>Soundex</encoding-type>
   </phoneticize-field>
</phoneticize-fields>
```

Master Index Phonetic Encoding Fields and Elements (Repository)

The following table lists and describes the Configuration Editor fields and XML file elements used to define how fields will be phonetically encoded.

Configuration Editor Field	XML File Element	Description
Unphoneticized Source	unphoneticized-source-field-name	The ePath of the source field in the system object that contains the data to be phonetically encoded (for example, Person.Address[*].StreetName). **Note –** This can refer to the original field or to a standardized or normalized field.
Phoneticized Target	phoneticized-target-field-name	The ePath of the field in the system object that will store the phonetically encoded value of the source field.

Configuration Editor Field	XML File Element	Description
	phoneticized-object-field-id	A field ID to identify the field to the phonetic encoder. This is not currently used with the Oracle Java CAPS Match Engine, but could be used with a custom standardization engine.
Encoder	encoding-type	The phonetic encoder to use for this field. This must correspond to an encoder defined in the Encoders section on the Configuration Editor (or the `PhoneticEncodersConfig` element of the XML file).

Modifying a Master Index Phonetic Encoding Definition (Repository)

Once you create a phonetic encoding definition, you can modify it as needed. Use caution when modifying definitions once a system is in production. This can cause inconsistent match results.

▼ To Modify a Phonetic Encoding Definition (Configuration Editor)

1. **In the Projects window, right-click the master index application you want to modify, and then click Open.**
2. **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

   The Configuration Editor appears.
3. **Click the Phoneticized Fields tab.**

   The Phoneticized Field page appears.
4. **In the Phoneticized Fields section, select the phonetic encoding definition you want to modify, and then click Edit.**

   The Phoneticized Field dialog box appears.
5. **Modify any of the fields listed in "Master Index Phonetic Encoding Fields and Elements (Repository)" on page 89, and then click OK.**
6. **On the Configuration Editor toolbar, click Save.**

▼ To Modify a Phonetic Encoding Definition (XML Editor)

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Match Field file.**

   The file opens in the NetBeans XML editor.

2 **Scroll to the `phoneticize-fields` element.**

3 **In the `phoneticize-field` element that defines the phonetic encoding you want to modify, change the value of any of the elements described in "Master Index Phonetic Encoding Fields and Elements (Repository)" on page 89.**

4 **Save and close the file.**

Deleting a Master Index Phonetic Encoding Definition (Repository)

Once you create a phonetic encoding definition, you can delete it as needed. Use caution when deleting definitions once a system is in production. This can cause inconsistent match results.

▼ To Delete a Phonetic Encoding Definition (Configuration Editor)

1 **In the Projects window, right-click the master index application you want to modify, and then click Open.**

2 **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

The Configuration Editor appears.

3 **Click the Phoneticized Fields tab.**

The Phoneticized Field page appears.

4 **In the Phoneticized Fields section, select the phonetic encoding definition you want to delete.**

5 **Click Remove.**

6 **Click Yes on the dialog that appears.**

7 **On the Configuration Editor toolbar, click Save.**

▼ To Delete a Phonetic Encoding Definition (XML Editor)

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Match Field file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `phoneticize-fields` element in the `PhoneticEncodersConfig` element.**

3 **Do either of the following:**

- **To delete a field currently specified for phonetic conversion, delete all text between and including the `phoneticize-field` element that defines the field.**

  Using the example below, to delete the first name phonetic field, delete the boldface text.

```
<phoneticize-fields>
   <phoneticize-field>
      <unphoneticized-source-field-name>Person.LastName
      </unphoneticized-source-field-name>
      <phoneticized-target-field-name>Person.LastNamePhoneticCode
      </phoneticized-target-field-name>
      <encoding-type>NYSIIS</encoding-type>
   </phoneticize-field>
   <phoneticize-field>
      <unphoneticized-source-field-name>Person.FirstName
      </unphoneticized-source-field-name>
      <phoneticized-target-field-name>Person.FirstNamePhoneticCode
      </phoneticized-target-field-name>
      <encoding-type>Soundex</encoding-type>
   </phoneticize-field>
</phoneticize-fields>
```

- **To delete all fields currently specified for phonetic conversion, delete all text between, but not including, the `phoneticize-fields` element.**

4 **Save and close the file.**

Defining a Master Index Phonetic Encoder (Repository)

Each type of phonetic encoder provided with Oracle Java CAPS Master Index is defined in the `PhoneticEncodersConfig` section of the Match Field file. They are listed in the Encoders section of the Configuration Editor.

▼ To Define a Phonetic Encoder (Configuration Editor)

1 **In the Projects window, right-click the master index application you want to modify, and then click Open.**

2 **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

The Configuration Editor appears.

3 **Click the Phoneticized Fields tab.**

The Phoneticized Field page appears.

4. **In the Encoders section, click Add.**

   The Phonetic Encoder dialog box appears.

5. **In the Encoder field, enter a descriptive name for the encoder.**

6. **In the Implementation Class field, enter the fully qualified Java path of the class to use for the encoder.**

   Note – For more information about the encoder class paths, see "Master Index Encoder Elements and Types (Repository)" on page 93.

7. **Click OK.**

   The phonetic encoder is added to the Encoders list.

8. **On the Configuration Editor toolbar, click Save.**

▼ To Define a Phonetic Encoder (XML Editor)

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Match Field file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `PhoneticEncodersConfig` section.**

3. **Create a new `encoder` element, and then define the elements described in "Master Index Encoder Elements and Types (Repository)" on page 93.**

4. **Save and close the file.**

Master Index Encoder Elements and Types (Repository)

The following table lists and describes the elements that configure the phonetic encoders used by the master index application.

Element	Description
encoding-type	The name of the phonetic encoder, such as NYSIIS, Soundex, or Metaphone. See the following table for a list of default encoders for the Oracle Java CAPS Match Engine.

Element	Description
encoder-implementation-class	The fully qualified name of the Java class that determines the behavior of the phonetic encoder. See the following table for a complete list of default classes for the Oracle Java CAPS Match Engine.

The following table lists the phonetic encoders supported by the Oracle Java CAPS Match Engine along with the names of their default classes.

Encoding Type	class-name
NYSIIS	com.stc.eindex.phonetic.impl.Nysiis
Soundex	com.stc.eindex.phonetic.impl.Soundex
Metaphone	com.stc.eindex.phonetic.impl.Metaphone
Double Metaphone	com.stc.eindex.phonetic.impl.DoubleMetaphone
Refined Soundex	com.stc.eindex.phonetic.impl.RefinedSoundex
French Soundex	com.stc.eindex.phonetic.impl.SoundexFR

Modifying a Master Index Phonetic Encoder (Repository)

Once you define a phonetic encoder, you can change the implementation class to use for the encoder. Use caution when modifying definitions once a system is in production. This can cause inconsistent match results.

▼ To Modify a Phonetic Encoder (Configuration Editor)

1. **In the Projects window, right-click the master index application you want to modify, and then click Open.**
2. **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

   The Configuration Editor appears.
3. **Click the Phoneticized Fields tab.**

   The Phoneticized Field page appears.
4. **In the Encoders section, select the encoder you want to modify and then click Edit.**

   The Phonetic Encoder dialog box appears.

5 **Change the implementation class, and then click OK.**

6 **On the Configuration Editor toolbar, click Save.**

▼ To Modify a Phonetic Encoder (XML Editor)

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Match Field file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `PhoneticEncodersConfig` section.**

3 **Modify the value of any of the elements in the `encoder` element you want to modify (for more information, see "Master Index Encoder Elements and Types (Repository)" on page 93.**

4 **Save and close the file.**

Deleting a Master Index Phonetic Encoder (Repository)

Once you define a phonetic encoder, you can delete it if needed. Use caution when deleting encoders once a system is in production. This can cause inconsistent match results. If you delete an encoder that is referenced by a phonetic encoding definition, make sure to modify that definition by referencing an existing encoder.

▼ To Delete a Phonetic Encoder (Configuration Editor)

1 **In the Projects window, right-click the master index application you want to modify, and then click Open.**

2 **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

The Configuration Editor appears.

3 **In the Configuration Editor toolbar, click the Phoneticized Fields tab.**

The Phoneticized Field page appears.

4 **In the Encoders section, select the encoder you want to delete.**

5 **Click Remove.**

6 **Click Yes on the dialog box that appears.**

7 **On the Configuration Editor toolbar, click Save.**

▼ To Delete a Phonetic Encoder (XML Editor)

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Match Field file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `PhoneticEncodersConfig` section.**

3 **Delete the text between and including the `encoder` element the defines the encoder you want to remove.**

4 **Save and close the file.**

Defining the Master Index Match String (Repository)

The match string defines the fields that are passed to the match engine for probabilistic weighting. By default, the fields defined for matching are the fields you specified for matching in the wizard or Configuration Editor. You can modify and delete fields in the match string if necessary. At least one field must be defined in the match string or no weights will be generated.

If you do modify the match string, you might need to make corresponding changes to the match engine configuration files as well. For more information about modifying these files, see the appropriate match engine implementation guide*Understanding the Oracle Java CAPS Match Engine*.

Perform either of the following tasks to configure the match string.

- "Creating the Master Index Match String (Repository)" on page 96
- "Modifying the Master Index Match String (Repository)" on page 98

Creating the Master Index Match String (Repository)

A default match string is predefined based on the match type information you specified in the wizard. If no match types were defined using the wizard, the structure of the match string is still defined but with no fields. You can use normalized or phonetically encoded fields for the match string.

To Create the Match String (Configuration Editor)

1. **In the Projects window, right-click the master index application you want to modify, and then click Open.**
2. **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

   The Configuration Editor appears.
3. **Expand the object structure so all fields are visible.**
4. **To add a field to the match string, click on the field name and then select a value for the Match Type field on the Properties page.**

   Note – The match types you can use are listed in the first column of `matchConfigFile.cfg`. For more information about Oracle Java CAPS Match Engine match types, see *Understanding the Oracle Java CAPS Match Engine*.
5. **Perform the previous step for each field in the match string.**
6. **On the Configuration Editor toolbar, click Save.**

To Create the Match String (XML Editor)

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Match Field file.**

   The file opens in the NetBeans XML editor.
2. **In the `MatchingConfig` element, scroll to the `match-columns` element in the `match-system-object` element.**
3. **To add a field to the match string, do the following:**

   a. **In the `match-columns` element, create a new `match-column` element.**

   b. **In the new `match-column` element, create and define a `column-name` element.**

      Enter the fully qualified field name of the field on which to match (for example, Enterprise.SystemSBR.Person.Address.City).

   c. **Following the `column-name` element, create and define a `match-type` element.**

      Enter an ID that identifies the field to the match engine. For the Oracle Java CAPS Match Engine, this value must correspond to a defined match type.

For example:

```
<match-system-object>
   <object-name>Address</object-name>
   <match-columns>
     <match-column>
       <column-name>Enterprise.SystemSBR.Person.Address.StreetName
       </column-name>
       <match-type>StreetName</match-type>
     </match-column>
   </match-columns>
</match-system-object>
```

4 **Repeat the previous step for each field to add to the match string.**

5 **Save and close the file.**

Modifying the Master Index Match String (Repository)

Once you define a match string, you can modify or delete information about the fields in the match string as necessary. This should only be done prior to moving to production. Otherwise, unexpected matching results might occur. For more information about Oracle Java CAPS Match Engine match types and field IDs, see *Understanding the Oracle Java CAPS Match Engine*.

▼ To Modify the Match String (Configuration Editor)

1 **In the Projects window, right-click the master index application you want to modify, and then click Open.**

2 **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

The Configuration Editor appears.

3 **Expand the object structure so all fields are visible.**

4 **To add a field to the match string, click the field name and then select a value for the Match Type field on the Properties page.**

The field is added to the match string.

5 **To modify the match type specified for a field, click the name of the field defined for matching and then select a new value for the Match Type field on the Properties page.**

6 **To remove a field from the match string, click the name of the field defined for matching and then select `None` for the Match Type field on the Properties page.**

7 **On the Configuration Editor toolbar, click Save.**

▼ To Modify the Match String (XML Editor)

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Match Field file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `MatchingConfig` element, and then scroll to the `match-system-object` element.**

3. **To add a field to the match string, do the following:**

   a. **In the `match-columns` element, create a new `match-column` element.**

   b. **In the new `match-column` element, create and define a `column-name` element.**

      Enter the fully qualified field name of the field on which to match (for example, Enterprise.SystemSBR.Person.Address.City).

   c. **Following the `column-name` element, create and define a `match-type` element.**

      Enter an ID that identifies the field to the match engine. For the Oracle Java CAPS Match Engine, this value must correspond to a defined match type.

      For example:

```
<match-system-object>
   <object-name>Address</object-name>
   <match-columns>
     <match-column>
       <column-name>Enterprise.SystemSBR.Person.Address.StreetName
       </column-name>
       <match-type>StreetName</match-type>
     </match-column>
   </match-columns>
</match-system-object>
```

4. **To change a field used in the match string, change the value of the `column-name` element.**

   Enter the fully qualified field name of the new field (for example, Enterprise.SystemSBR.Person.FirstName).

5. **To change the type of matching to perform for a field, change the value of the `match-type` element.**

   Enter an ID that identifies the field to the match engine. For the Oracle Java CAPS Match Engine, this value must correspond to a defined match type.

6 **To delete a field from the match string, delete all text between and including the `match-column` element defining the field you want to delete.**

Using the example below, to delete the HouseNo field from the match string, delete the boldface text.

```
<match-system-object>
  <object-name>Address</object-name>
  <match-columns>
     <match-column>
       <column-name>Enterprise.SystemSBR.Person.Address.StreetName
       </column-name>
       <match-type>StreetName</match-type>
     </match-column>
     <match-column>
       <column-name>Enterprise.SystemSBR.Person.Address.HouseNo
       </column-name>
       <match-type>HouseNumber</match-type>
     </match-column>
```

7 **Save and close the file.**

Defining how Master Index Query Blocks are Processed (Repository)

The block picker and pass controller define how query blocks are processed for matching. Default components are defined by Oracle Java CAPS Master Index, but you can create your own Java classes to define custom versions of these components.

The block picker determines the blocking strategy to use for each match pass. Blocking strategies define how the queries are created that check the database for a subset of the records to be used for matching. The default Block Picker has access to the match results from previous match passes, as well as lists of applicable blocking definitions that have been executed and of those that have not. The default Block Picker class is `com.stc.eindex.matching.impl.PickAllBlocksAtOnce`, which selects all blocks during the first pass.

The pass controller determines whether to continue processing the defined blocks. The matching process can be executed in multiple stages. Each query block in the blocking query is executed in a separate match pass. After a block is evaluated, the pass controller determines if the results found are sufficient or if the query should continue by performing another match pass. The default pass controller is `com.stc.eindex.matching.impl.PassAllBlocks`. This class instructs the match engine to continue calculating match weights until all applicable block definitions have been processed.

These components can only be configured by modifying the XML file directly.

To Specify the Block Picker

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Match Field file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `block-picker` element in the `MatchingConfig` section.**

3. **In the `class-name` element, specify the Java class for the block picker you want to use, using the fully qualified class name.**

   For example:

   ```
   <block-picker>
      <class-name>com.stc.eindex.matching.impl.PickAllBlocksAtOnce
      </class-name>
   </block-picker>
   ```

4. **Save and close the file.**

To Specify the Pass Controller Class

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Match Field file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `pass-controller` element in the `MatchingConfig` section.**

3. **In the `class-name` element, specify the Java class for the Pass Controller you want to use, using the fully qualified class name.**

   For example:

   ```
   <pass-controller>
      <class-name>com.stc.eindex.matching.impl.PassAllBlocks
      </class-name>
   </pass-controller>
   ```

4. **Save and close the file.**

Defining the Master Index Survivor Calculator (Repository)

The survivor calculator is configured in the Best Record file and defines how field values are populated into the single best record, along with the type of survivor strategy to use for each field. If no strategy is defined for a field, the default survivor strategy is used to populate that field in the SBR. Before you begin to define the survivor calculator, make sure you know which fields to include in the SBR and how those fields should be updated. Typically, all but phonetic and standardized fields are included here.

The survivor calculator can only be configured by modifying the XML file directly. Perform the following tasks to configure the survivor calculator.

Specifying the Master Index Survivor Helper (Repository)

The survivor helper class determines how to retrieve values from system records and how to set them in the SBR. The default class is `com.stc.eindex.survivor.impl.DefaultSurvivorHelper`, which uses the ePath method to retrieve and set the values. You can create a custom survivor helper class to support other methods for retrieving and setting values. If you implement a custom survivor helper class, it must extend`com.stc.eindex.survivor.AbstractSurvivorHelper`.

▼ To Specify the Survivor Helper

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Best Record file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `helper-class` element in the `SurvivorHelperConfig` element.**

3. **Change the value of the `helper-class` element to the fully qualified name of new helper class.**

   For example:

   ```
   <helper-class>com.stc.eindex.survivor.impl.MySurvivorHelper
   </helper-class>
   ```

4. **Save and close the file.**

Specifying a Master Index Default Survivor Strategy (Repository)

The default survivor strategy specifies the name of the Java class that defines the survivor calculation strategy to use for most of the fields in the SBR. By defining a default strategy, you do not need to define a strategy for every candidate field; you only need to define a strategy for fields that do not use the default strategy.

Note – If you create a customized class for the default survivor strategy, make sure the class implements `com.stc.eindex.survivor.SurvivorStrategyInterface` and is accessible by the EJB class loader.

▼ To Specify a Default Survivor Strategy

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Best Record file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `default-survivor-strategy` element in the `SurvivorHelperConfig` element.**

3. **To change the name of the default class, modify the value of the `strategy-class` element to the fully qualified name of the new default strategy class.**

   For example:

   ```
   <default-survivor-strategy>
      <strategy-class>com.stc.eindex.survivor.impl.MySurvivorStrategy
      </strategy-class>
   </default-survivor-strategy>
   ```

4. **Configure the strategy parameters, as described in "Configuring the Default Survivor Strategy" on page 103.**

5. **Save and close the file.**

Configuring the Default Survivor Strategy

Once you define a default survivor strategy, you might need to specify certain parameters for the strategy. One parameter is required for the WeightedSurvivorStrategy and for the DefaultSurvivorStrategy. If you create a custom strategy, additional parameters can be used.

▼ To Configure the Default Survivor Strategy

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Best Record file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `default-survivor-strategy` element, and then to the `strategy-class` element.**

3 **Do any of the following:**

- **To modify an existing parameter value, scroll to the parameter you want to modify, and then change the value of the `parameter-value` element.**

  For example:

```
<default-survivor-strategy>
   <strategy-class>com.stc.eindex.survivor.impl.MySurvivorStrategy
   </strategy-class>
   <parameters>
      <parameter>
         <parameter-name>ConfigModuleName</parameter-name>
         <parameter-type>java.lang.String</parameter-type>
         <parameter-value>MySurvivorCalculator</parameter-value>
      </parameter>
   </parameters>
</default-survivor-strategy>
```

- **To add a new parameter, create a new `parameter` element within the `parameters` element, and then define the parameter elements described in "Master Index Default Survivor Strategy Parameter Elements (Repository)" on page 104.**

- **To delete a parameter, scroll to the `parameters` element, and then delete all text between and including the `parameter` element that define the parameter.**

- **To delete all parameters, delete the all text between and including the `parameters` element.**

4 **Save and close the file.**

Master Index Default Survivor Strategy Parameter Elements (Repository)

The following table lists and describes the elements that configure the parameters for the default survivor strategy in the Best Record file.

Element	Value
description	This is an optional element that briefly describes the parameter. Note that there are no parameters to define for the `UnionSurvivorStrategy`.
parameter-name	The name of the parameter. ■ For the DefaultSurvivorStrategy, this value is "preferredSystem". ■ For the WeightedSurvivorStrategy, this value is "ConfigurationModuleName". ■ For the UnionSurvivorStrategy, this is not used.
parameter-type	The Java data type for the parameter value. For both the DefaultSurvivorStrategy and the WeightedSurvivorStrategy, this value is `java.lang.String`.
parameter-value	The value of the named parameter. ■ For the DefaultSurvivorStrategy, this is the processing code of the source system from which the SBR field value is retrieved. ■ For the WeightedSurvivorStrategy, this is the name of the `module-name` element that defines the weighted calculator to use as the default strategy (by default, WeightedSurvivorCalculator).

Defining the Master Index Single Best Record Structure (Repository)

In order for a field to be populated in the SBR, that field must be defined in the candidate field list of the survivor helper. By default, all the fields that were specified in the wizard are also defined here. Any candidate fields defined for the SBR must also be defined in the Object Definition file. If you add a field to the object definition, you should also add the field here.

The SBR can only be defined by modifying the XML file directly.

▼ To Specify a Candidate Field

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Best Record file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `candidate-definitions` element in the `SurvivorHelperConfig` element.**

3 **Do any of the following:**

- **To add a new field, add a new `candidate-field` element within the `candidate-definitions` tags, and then name the new element using the ePath of the field.**

  For example:

  ```
  <candidate-definitions>
     <candidate-field name="Person.LastName"/>
     <candidate-field name="Person.DateOfBirth"/>
  <candidate-definitions>
  ```

- **To modify an existing field, scroll to the `candidate-field` element you want to modify, and then change the value of the attribute.**

- **If any of the updated fields do not use the default strategy, define a strategy for those fields, as described in "Defining a Master Index Survivor Strategy for a Field or Object (Repository)" on page 107.**

4 **Save and close the file.**

Deleting Candidate Fields

Once a field is defined in the SBR candidate field list, you can delete the field if you do not want to include the field in the SBR. If you delete a field from the object definition, make sure to delete the field here as well.

▼ To Delete a Candidate Field

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Best Record file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `candidate-definitions` element in the `SurvivorHelperConfig` element.**

3 **Delete all text between and including the appropriate `candidate-field` element.**

Using the example below, to delete the Religion field, delete the boldface text; to delete the Alias object, delete the plain text.

```
<candidate-field name="Person.Religion"/>
<candidate-field name="Person.Alias[*].*"/>
   <system-fields>
      <field-name>LastModified</field-name>
   </system-fields>
   <survivor-strategy>
      <strategy-class>com.stc.eindex.user.MyStrategy
      </strategy-class>
   </survivor-strategy>
</candidate-field>
```

Note – Do not delete all candidate fields; at a minimum, the match fields must be defined.

4 **Save and close the file.**

Defining a Master Index Survivor Strategy for a Field or Object (Repository)

To use a strategy for a specific field other than the strategy defined in the `default-survivor-strategy` element, you need to specify the new strategy for the appropriate `candidate-field` element. A `candidate-field` element can represent a field or child object. You do not need to specify a strategy for any fields that use the default survivor strategy.

▼ To Define a Survivor Strategy for a Field

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Best Record file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `candidate-definitions` element in the `SurvivorHelperConfig` element.**

3 **In the `candidate-field` element for which you want to specify a new strategy, create a new `survivor-strategy` element.**

For example:

```
<candidate-field name="Person.Alias[*].*">
   <survivor-strategy>
   </survivor-strategy>
</candidate-field>
```

4 **In the new `survivor-strategy` element, create and name a new `strategy-class` element.**

Make sure to specify the fully qualified name of the Java class for the strategy. For example:

```
<candidate-field name="Person.Alias[*].*">
   <survivor-strategy>
      <strategy-class>
       com.stc.eindex.survivor.impl.UnionSurvivorStrategy
      </strategy-class>
   </survivor-strategy>
</candidate-field>
```

Note – To specify the default survivor strategy for a field, make sure the corresponding `candidate-field` element does not contain a `survivor-strategy` element. If you implement a custom strategy class, that class must be defined as a custom plug-in.

5 Save and close the file.

Defining Master Index Custom Weighted Strategies (Repository)

The `WeightedCalculator` element defines the Java class used for weighted survivor calculations. If the weighted calculator is defined for the `default-survivor-strategy` element, then the strategies you define here are used for all fields for which no specific survivor strategy is defined. The weighted calculator defines a default strategy to use for most fields and specialized strategies to use for specific fields.

Configuring the weighted calculator involves the following tasks.

Defining Custom Weighted Strategies

The `WeightedCalculator` element defines both default and custom survivor strategies. You can override the default weighted calculator strategy for specific fields by defining custom strategies for those fields in the `candidate-field` elements of the `WeightedCalculator` element.

▼ To Define Custom Weighted Calculators

1 In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Best Record file.

The file opens in the NetBeans XML editor.

2 Scroll to the `WeightedCalculator` element.

3 Add and name a new `candidate-field` element in the `WeightedCalculator`.

For example:

```
<WeightedCalculator module-name="WeightedSurvivorCalculator"
 parser-class=
 "com.stc.eindex.configurator.impl.WeightedCalculatorConfig">
   <candidate-field name="Person.DOB">
   </candidate-field>
```

4. **Create one or more `parameter` elements for the new candidate field.**

```
<candidate-field name="Person.DOB">
   <parameter>
   </parameter>
   <parameter>
   </parameter>
</candidate-field>
```

5. **For each new `parameter` element, define the elements listed in "Master Index Weighted Calculator Parameter Elements (Repository)" on page 112.**

```
<candidate-field name="Person.DOB">
   <parameter>
      <quality>SourceSystem</quality>
      <preference>CDI</preference>
      <utility>80.0</utility>
   </parameter>
   <parameter>
      <quality>MostRecentModified</quality>
      <utility>75.0</utility>
   </parameter>
</candidate-field>
```

6. **Save and close the file.**

Configuring Weighted Strategies

The wizard creates a default weighted strategy that defines a general weighting structure to be used by most fields. Unless custom weighted calculator strategies are defined for a field, the default strategies defined in the `default-parameters` element are used for each field using the weighted calculator.

▼ To Add Default Weighted Calculator Parameters

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Best Record file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `default-parameters` element in the `WeightedCalculator` element.**

3. **Create new a new `parameter` element within the `default-parameters` element, but outside any existing `parameter` elements.**

4. **In the new `parameter` element, define the elements listed in "Master Index Weighted Calculator Parameter Elements (Repository)" on page 112.**

   For example:

```
<default-parameters>
   <parameter>
      <quality>SourceSystem</quality>
```

```
      <preference>CDA</preference>
      <utility>80.0</utility>
   </parameter>
   <parameter>
      <quality>MostRecentModified</quality>
      <utility>75.0</utility>
   </parameter>
</default-parameters>
```

5 **Save and close the file.**

Modifying Weighted Calculator Parameters

Once a candidate field is specified and custom weighted calculators are defined for the field, you can modify the parameters. You can also modify any existing default weighted calculator parameters.

▼ To Modify Weighted Calculator Parameters

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Best Record file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `WeightedCalculator` element.**

3 **Do either of the following:**

- **To modify a custom weighted calculator parameter, scroll to the `candidate-field` element naming the field to modify.**
- **To modify a default weighted calculator parameter, scroll to the `default-parameters` element.**

4 **Modify the value of any of the elements listed in "Master Index Weighted Calculator Parameter Elements (Repository)" on page 112.**

For example:

```
<parameter>
   <quality>SourceSystem</quality>
   <preference>DDI</preference>
   <utility>60.0</utility>
</parameter>
```

5 **Save and close the file.**

Deleting Weighted Calculator Parameters

Once default and custom parameters are defined, they can be deleted if necessary. If a candidate field is defined for custom weighted calculations, you can specify that the field use the default weighted calculator instead by removing the entire field from the `candidate-fields` list.

▼ To Delete Weighted Calculator Parameters

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Best Record file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `WeightedCalculator` element and then to the `candidate-field` element identifying the field you want to delete.**

3. **Do either of the following:**

   - **To delete a custom weighted calculator parameter, scroll to the `candidate-field` element naming the field to modify.**
   - **To delete a default weighted calculator parameter, scroll to the `default-parameters` element.**

4. **To delete an existing parameter, scroll to the `parameter` element you want to delete, and then remove all text between and including the `parameter` element.**

   For example, to delete the SourceSystem parameter below, delete the boldface text.

```
<parameter>
   <quality>SourceSystem</quality>
   <preference>CDI</preference>
   <utility>80.0</utility>
</parameter>
<parameter>
   <quality>MostRecentModified</quality>
   <utility>75.0</utility>
</parameter>
```

Note – At least one parameter must be defined for the `default-parameter` element; you cannot delete all parameters from this section. You cannot delete all parameters from a candidate field, but you can delete the entire candidate field (see below for more information).

---

5 **To delete a field from the candidate field list, delete all text between and including the `candidate-field` tags for the field you want to delete.**

Using the following example, to delete the Person.DOB candidate field from the custom calculator, delete all the text below.

```
<candidate-field name="Person.DOB">
   <parameter>
      <quality>SourceSystem</quality>
      <preference>CDI</preference>
      <utility>80.0</utility>
   </parameter>
   <parameter>
      <quality>MostRecentModified</quality>
      <utility>75.0</utility>
   </parameter>
</candidate-field>
```

6 **Save and close the file.**

Master Index Weighted Calculator Parameter Elements (Repository)

The following table lists and describes the elements that configure the parameters for the weighted calculator in the Best Record file.

Element	Description
quality	The type of weighted calculation to perform, such as: ■ SourceSystem ■ SystemAgreement ■ MostRecentModified For more information about these qualities, see "Best Record Configuration (Repository)" in *Understanding Oracle Java CAPS Master Index Configuration Options (Repository)*.
preference	The preferred value for the specified quality. This element is only used for the SourceSystem quality and the preference must be a source system code.
utility	A value that indicates the reliability of the specified quality for determining the best field value for the SBR. You define the scale for the utility values.

Configuring Master Index Update Policies (Repository)

When the Best Record file is generated, no Java classes are defined for the update policies. You can create custom update policy classes and specify that the custom classes be used instead. Custom update policies must implement `com.stc.eindex.update.UpdatePolicy` and must be defined as custom classes in the Oracle Java CAPS Master Index project to be recognized as an update policy. The names of the custom plug-ins you create are the values you enter for the update policies. You can also set the update policy flag to specify whether the policies are performed when no changes are made to an existing record.

Note – Master Patient Index includes update policies to assist with alias name processing. These policies are defined in the Best Record file by default for the Master Patient Index project.

Perform the following tasks to configure the update policies:

- "Defining Master Index Update Policies (Repository)" on page 113
- "Setting the Master Index Update Policy Flag (Repository)" on page 114

Defining Master Index Update Policies (Repository)

You can define update policies for any of the seven update policy elements. You do not need to specify a policy for each element.

▼ To Define Update Policies

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Best Record file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `UpdateManagerConfig` section of the file.**

3. **Do any of the following:**

   - **To modify the merge policy for enterprise objects, change the value of the `EnterpriseMergePolicy` element to the fully qualified name of the new Java class.**

     For example:

     ```
     <EnterpriseMergePolicy>com.stc.eindex.user.MyEntMergePolicy
     </EnterpriseMergePolicy>
     ```

- **To modify the unmerge policy for enterprise objects, change the value of the `EnterpriseUnmergePolicy` element to the fully qualified name of the new Java class.**

  For example:

  ```
  <EnterpriseUnmergePolicy>com.stc.eindex.user.MyEntUnmergePolicy
  </EnterpriseUnmergePolicy>
  ```

- **To modify the update policy for enterprise objects, change the value of the `EnterpriseUpdatePolicy` element to the fully qualified name of the new Java class.**

  For example:

  ```
  <EnterpriseUpdatePolicy>com.stc.eindex.user.MyEntUpdatePolicy
  </EnterpriseUpdatePolicy>
  ```

- **To modify the create policy for enterprise objects, change the value of the `EnterpriseCreatePolicy` element to the fully qualified name of the new Java class.**

  For example:

  ```
  <EnterpriseCreatePolicy>com.stc.eindex.user.MyCreatePolicy
  </EnterpriseCreatePolicy>
  ```

- **To modify the merge policy for system objects, change the value of the `SystemMergePolicy` element to the fully qualified name of the new merge policy Java class.**

  For example:

  ```
  <SystemMergePolicy>com.stc.eindex.user.MySysMergePolicy
  </SystemMergePolicy>
  ```

- **To modify the unmerge policy for system objects, change the value of the `SystemUnmergePolicy` element to the fully qualified name of the new Java class.**

  For example:

  ```
  <SystemUnmergePolicy>com.stc.eindex.user.MySysUnmergePolicy
  </SystemUnmergePolicy>
  ```

- **To modify the assumed match policy, change the value of the `UndoAssumeMatchPolicy` element to the fully qualified name of the new Java class.**

  For example:

  ```
  <UndoAssumeMatchPolicy>com.stc.eindex.user.MyUndoAsmMatchPolicy
    </UndoAssumeMatchPolicy>
  ```

4. **Save and close the file.**

Setting the Master Index Update Policy Flag (Repository)

The update flag determines whether update policies are performed against a record when a transaction does not cause any changes to the record's data.

▼ To Set the Update Policy Flag

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Best Record file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `UpdateManagerConfig` section of the file.**

3. **To specify that update policies are not performed when no updates are made, set the `SkipUpdateIfNoChange` element to `true`.**

   For example:

   ```
   <SkipUpdateIfNoChange>true</SkipUpdateIfNoChange>
   ```

4. **To specify that update policies are performed even though no updates are made, set the `SkipUpdateIfNoChange` element to `false`.**

   For example:

   ```
   <SkipUpdateIfNoChange>false</SkipUpdateIfNoChange>
   ```

5. **Save and close the file.**

Defining Custom Field Validations for the Master Index (Repository)

By default, the Field Validation file defines one validation rule named "validate-local-id". This rule defines certain validations to perform against local ID and system fields before they are entered into the database. The local ID validator verifies that the system code is valid, the local ID format is correct, the local ID is the correct length, and that neither field is null.

You can define additional validations for your data by creating the Java class and appropriate methods. Create the custom class using the Custom Plug-ins module of the master index project so you can include the custom validation rule in the master index application code package. The custom validation classes must implement `com.stc.eindex.objects.validation.ObjectValidator`. The exception thrown is `com.stc.eindex.objects.validation.exception.ValidationException`.

▼ To Implement a Validation Rule

1. **Create the custom validation rules using the Custom Plug-in feature.**

2. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Field Validation file.**

   The file opens in the NetBeans XML editor.

3. **Create a new `rule` element in the `rules` element.**

4. **In the new rule element, create the following attributes:**

   - **`name` – A unique name for the validation rule.**
   - **`object-name` – The name of the master index Java class that defines the object to which the validation rule is applied, such as `SystemObject` or `Parent_NameObject` (where *Parent_Name* is the name of the parent object in the object definition)**
   - **`class` – The complete path of the Java class to call for the validation rule.**

     For example:

```
<ValidationConfig module-name="Validation" parser-class=
 "com.stc.eindex.configurator.impl.validation.ValidationConfiguration"
   <rules>
      <rule name="validate-auxiliary-id" object-name="PersonObject"
       class="com.stc.eindex.user.AuxiliaryId"/>
      <rule name="validate-birth-date" object-name="PersonObject"
       class="com.stc.eindex.user.BirthDate"/>
   </rules>
</ValidationConfig>
```

5. **Save and close the file.**

Configuring the Match Engine (Repository)

You can configure the match engine by specifying the match engine to use and configuring the predefined comparison functions. You only need to specify the match engine to use if you are using an engine other than the Oracle Java CAPS Match Engine.

Perform any of these steps to configure the match engine:

- "Specifying a Match Engine for the Master Index (Repository)" on page 117
- "Configuring the Comparison Functions for a Master Index Application (Repository)" on page 118

Specifying a Match Engine for the Master Index (Repository)

Oracle Java CAPS Master Index can support different match engines depending on the adapter configured to communicate with the engine. Default classes are provided for using the Oracle Java CAPS Match Engine. You can implement a custom match engine along with custom adapters. The match engine configuration is defined by the `matcher-api` and `matcher-config` elements.

Note – The default adapters for the Oracle Java CAPS Match Engine are `com.stc.eindex.matching.adapter.SbmeMatcherAdapter` and `com.stc.eindex.matching.adapter.SbmeMatcherAdapterConfig`.

▼ To Configure the Match Engine

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Match Field file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `matcher-api` element in the `MatchingConfig` section.**

3. **Specify the Java class for the matching adapter to use, using the fully qualified class name as shown below.**

```
<matcher-api>
   <class-name>com.stc.eindex.matching.adapter.SbmeMatcherAdapter
   </class-name>
</matcher-api>
```

4. **In the `matcher-config` element, specify the Java class for the configuration of the matching adapter, using the fully qualified class name as shown below.**

```
<matcher-config>
   <class-name>
    com.stc.eindex.matching.adapter.SbmeMatcherAdapterConfig
   </class-name>
</matcher-config>
```

5. **Save and close the file.**

Configuring the Comparison Functions for a Master Index Application (Repository)

The match configuration file in the Match Engine node of the master index project lists and defines the configuration for each match type based on the predefined comparison function for the Oracle Java CAPS Match Engine. These match types can be applied to each field in the match string. You can modify the configuration of the existing matches types, add new match types, and specify whether the match engine should use agreement and disagreement weights or m-probabilities and u-probabilities.

For more information about the structure of the match configuration file and the comparison functions you can use, see *Understanding the Oracle Java CAPS Match Engine*.

▼ To Configure the Comparison Functions (Configuration Editor)

1. **In the Projects window, right-click the master index application you want to modify, and then click Open.**

2. **If the Configuration Editor dialog box appears, click Edit to check out the listed files.**

   The Configuration Editor appears.

3. **Click the Matching tab.**

   The Matching page appears with a list of fields defined for matching and a list of comparators that you can modify.

4. **In the Probability Type field, select one of the following:**

   - **Use Agree/Disagreement Weight Ranges** – Uses agreement and disagreement weights for matching. If agreement and disagreement weights are used, the m-probability and u-probability fields are ignored and do not appear on the Matching page.
   - **Use M-Probabilities/U-Probabilities** – Uses m-probabilities and u-probabilities for matching. If m-probabilities and u-probabilities are used, the agreement and disagreement weight fields are ignored and do not appear on the page.

5. **To add a new matching rule:**

   a. **Click Add in the lower right portion of the window.**

      The Edit Matching Rules dialog box appears.

   b. **Fill in the fields described in "Match Comparator Configuration Properties for Oracle Java CAPS Master Index (Repository)" on page 120.**

   c. **Click OK.**

6. **To edit an existing matching rule:**

   a. **Click Edit in the lower right portion of the window.**

      The Edit Matching Rules dialog box appears.

   b. **Change the value of any of the fields described in "Match Comparator Configuration Properties for Oracle Java CAPS Master Index (Repository)" on page 120.**

   c. **Click OK.**

7. **To remove an existing matching rule:**

   a. **In the matching rule table, select the rule you want to delete.**

   b. **Click Remove.**

8. **On the Configuration Editor toolbar, click Save.**

▼ To Configure the Comparison Functions (Text Editor)

1. **In the project window, expand the master index project, and then expand the master index application.**

2. **In the Match Engine folder, double-click `matchConfigFile.cfg`.**

3. **For the Probability Type, enter one of the following values:**

   - **0** – Uses m-probabilities and u-probabilities for matching. If m-probabilities and u-probabilities are used, the agreement and disagreement weight fields are ignored.
   - **1** – Uses agreement and disagreement weights for matching. If agreement and disagreement weights are used, the m-probability and u-probability fields are ignored.

4. **For each comparison function you want to configure, modify the value of any of the columns described in "Match Comparator Configuration Properties for Oracle Java CAPS Master Index (Repository)" on page 120.**

5. **Save and close the file.**

Match Comparator Configuration Properties for Oracle Java CAPS Master Index (Repository)

The following table lists and describes the Configuration Editor fields used to define the comparison functions. It also lists the corresponding column in the match configuration file if you want to modify the file directly.

Configuration Editor Field	Match Configuration File Element and Column Number	Description
Match Type	match-type (column 1)	A value that indicates to the Oracle Java CAPS Match Engine how each field should be weighted. Each field included in the match string (the `MatchingConfig` section of the Match Field file) must have a match type corresponding to a match type defined in this file.
Match Size	size (column 2)	The number of characters on which matching is performed, beginning with the first character. For example, to match on only the first four characters in a 10-digit field, the value of this column should be "4".

Configuration Editor Field	Match Configuration File Element and Column Number	Description
Null Field	null-field (column 3)	An index that specifies how to calculate the total weight for null fields or fields that only contain spaces. You can specify any of the following values. The Configuration Editor value is given first, followed by the match configuration file value in parenthesis. ■ **Zero weight (0)** - If one or both fields are empty, the weight used for the field is 0 (zero). ■ **Full Combination weight (1)** - If both fields are empty, the agreement weight is used; if only one field is empty, the disagreement weight is used. ■ **Full Agreement weight (a1)** - Specifies to use the full agreement weight if both fields are null. ■ **1/x of the Agreement weight (ax)** - Specifies to use the a fraction of the agreement weight if both fields are empty. The agreement weight is multiplied by the fraction 1/x to obtain the match weight for that field. When modifying the match configuration file directly, the default is "2" if no number is specified. You can specify any number from 1 through 10. ■ **Full disagreement weight (d1)** - Specifies to use the full disagreement weight if both fields are null. ■ **1/x of the disagreement weight (dx)** - Specifies to use the disagreement weight if only one field is empty. The disagreement weight is multiplied by the fraction 1/x to obtain the match weight for the field. When modifying the match configuration file directly, the default is "2" if no number is specified. You can specify any number from 1 through 10. In the above descriptions, the agreement and disagreement weights are either specified in this file or calculated using a logarithmic formula based on the m and u-probabilities (depending on the probability type).

Configuration Editor Field	Match Configuration File Element and Column Number	Description
Function	function (column 4)	The type of comparison to perform when weighting the field. For information about the available comparison functions, see "Match Configuration Comparison Functions for Oracle Java CAPS Match Engine (Repository)" in *Understanding the Oracle Java CAPS Match Engine*.
Agreement Weight	agreement-weight (column 7)	The matching weight to be assigned to a field given that the fields match between two records; that is, the maximum match weight for a field. This number can be between 0 and 100 and can have up to 16 decimal points. Only set this value if the Probability Type is set to use agreement and disagreement weights.
Disagreement Weight	disagreement-weight (column 8)	The matching weight to be assigned to a field given that the fields do not match between two records; that is, the minimum match weight for a field. This number can be between 0 and -100 and can have up to 16 decimal points. Only set this value if the Probability Type is set to use agreement and disagreement weights.
M-Probability	m-prob (column 5)	The initial probability that the specified field in two records will match if the records match. The probability is a double value between 0 and 1, and can have up to 16 decimal points. Only set this value if the Probability Type is set to use probabilities.
U-Probability	u-prob (column 6)	The initial probability that the specified field in two records will match if the records do not match. The probability is a double value between 0 and 1, and can have up to 16 decimal points. Only set this value if the Probability Type is set to use probabilities.
Extra Parameters	parameters (column 9)	Parameters correspond to the comparison function specified in the Function field or column. Some comparison functions do not take any parameters and some take multiple parameters. For information about which functions take parameters and the parameters they take, see "Match Configuration Comparison Functions for Oracle Java CAPS Match Engine (Repository)" in *Understanding the Oracle Java CAPS Match Engine*.

Configuring the Standardization Engine (Repository)

You can configure the standardization engine by specifying the standardization engine to use, configuring the files that define data standardization, and loading standardization files into the master index application. You only need to specify the standardization engine to use if you are using an engine other than the Oracle Java CAPS Match Engine.

Perform any of these steps to configure the standardization engine:

- "Specifying a Standardization Engine for the Master Index (Repository)" on page 123
- "Modifying Master Index Standardization Files (Repository)" on page 124
- "Loading Standardization Files to a Master Index Application (Repository)" on page 124

Specifying a Standardization Engine for the Master Index (Repository)

Oracle Java CAPS Master Index can support standardization engines from different vendors depending on the adapter configured to communicate with the engine. Default classes are provided for using the Oracle Java CAPS Match Engine. You can implement a custom standardization engine along with customized adapters. The standardization engine configuration is defined by `standardizer-api` and `standardizer-config` elements.

Note – The default adapters for the Oracle Java CAPS Match Engine are `com.stc.eindex.matching.adapter.SbmeStandardizerAdapter` and `com.stc.eindex.matching.adapter.SbmeStandardizerAdapterConfig`.

▼ To Specify the Standardization Engine

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Match Field file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `standardizer-api` element in the `MatchingConfig` section.**

3. **Specify the Java class for the standardization adapter to use, using the fully qualified class name as shown below.**

```
<standardizer-api>
   <class-name>
    com.stc.eindex.matching.adapter.MyStandardizerAdapter
   </class-name>
</standardizer-api>
```

4. **In the `standardizer-config` element, specify the Java class for the configuration of the standardization adapter, using the fully qualified class name as shown below.**

```
<standardizer-config>
   <class-name>
     com.stc.eindex.matching.adapter.SbmeStandardizerAdapterConfig
   </class-name>
</standardizer-config>
```

5. **Save and close the file.**

Modifying Master Index Standardization Files (Repository)

You can fine-tune the standardization process by modifying the standardization files. For example, you can insert additional names or terms into the nickname file or business name files. Depending on your data requirements, you might need to modify additional standardization files. Some of the patterns files (most notably the address patterns files) are very complex and should only be modified by personnel who thoroughly understand the defined patterns and tokens. If you modify standardization files, make sure you modify them for each national domain specified in the Match Field file.

▼ To Modify Standardization Data Configuration Files

1. **In the Projects window, expand the master index project to configure and then expand Standardization Engine.**
2. **If the file you want to modify is domain-specific, expand the domain name.**
3. **Open the file you want to modify in the NetBeans text editor.**
4. **Modify the file in accordance with the information presented for each data type in** *Understanding the Oracle Java CAPS Match Engine*.
5. **Save and close the file.**

Loading Standardization Files to a Master Index Application (Repository)

Loading the standardization files brings them into the Repository and the master index project. This procedure is only required for projects that were upgraded from previous versions and that do not contain all the needed files. The files are loaded into the Standardization Engine node,

with domain-specific files being loaded into their own subdirectory. In a fresh installation of Oracle Java CAPS Master Index, all files are automatically loaded.

▼ To Load Standardization Files

1. **In the Project window, expand the master index project, and then expand the master index application.**
2. **Right-click the Standardization Engine folder, and then select Load Configuration Files from the context menu.**
3. **In the Open dialog box, open the folder containing the files you want to load.**
4. **Select the files to load, and then click Open.**
5. **On the Information dialog box, click OK.**

Configuring the Master Index EDM Appearance (Repository)

You can modify the configuration of the fields and objects to specify how they appear on the EDM pages by modifying the Enterprise Data Manager file. You can perform any of the following actions to customize the general appearance of the EDM. Though some appearance options can be configured using the Configuration Editor, it is best to modify the XML file directly.

Perform any of the following tasks to configure the appearance of the EDM.

Adding Objects to the EDM

You can define additional objects for the EDM as long as those objects are defined in the Object Definition file. Each object can only contain the fields that are also defined for that object in the Object Definition file. If you add objects to the object structure using the Configuration Editor, the new objects are automatically added to the the Enterprise Data Manager file and you do not need to perform these steps.

▼ To Add an Object to the EDM

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

The file opens in the NetBeans XML editor.

2 **In the nodes list of the file, create a new `node-`*object* element, substituting the object name for *object*, as shown below.**

```
<node-Phone>
</node-Phone>
```

3 **For any child objects, define the following attributes.**

- `display-order` – The order in which the child object types are displayed in the tree view pane on the EDM pages.
- `merge-must-delete` – An indicator of whether EDM users must specify which the instances of the child object type to retain during a system record merge. Specify **`false`** to allow EDM users to select the child objects to retain or to accept the default objects (the default objects are those in the destination system). Specify **`true`** to force the user to select the child objects to retain. This allows you to hide certain child object types on the EDM while forcing the user to specify which visible child objects to retain.

  For example:

  ```
  <node-Phone display-order="3" merge-must-delete="false">
  </node-Phone>
  ```

4 **Define fields for the new object, as described in "Adding Fields to the EDM" on page 128.**

5 **Define the relationship of the object, as described in "Defining EDM Object Relationships (Repository)" on page 136.**

6 **Save and close the file.**

Modifying EDM Objects

Once an object is defined in the Enterprise Data Manager file, you can modify the name or display order. Only modify the name of an object if you modify the corresponding object name in the Object Definition file and the remaining configuration files. Do not change the parent object name. If you modify an object in the object definition on the Configuration Editor, the the Enterprise Data Manager file is automatically updated as well.

▼ To Modify an EDM Object

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `node` element naming the object to modify.**

3. **To modify the object name, change the value of the `node` element to `node-`*object_name*, where *object_name* is the new name for the object.**

4. **To modify the location of the object in the EDM tree structure, change the value of the `display-order` attribute.**

   For example:

   ```
   <node-Alias display-order="2">
   ```

5. **If necessary, renumber the order of other child objects so they are sequential.**

6. **Save and close the file.**

Deleting Objects From the EDM

Once an object is defined in the Enterprise Data Manager file, you can remove the object. If the object remains defined in the Object Definition file, then the object is still a part of the enterprise record, but does not appear on the EDM. Before removing an object from the the Enterprise Data Manager file, make sure none of its fields are required in order to create a new record. If you delete an object using the Configuration Editor, the object is automatically deleted from the Enterprise Data Manager file.

▼ To Delete an Object

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

   The file opens in the NetBeans XML editor.

2 **Scroll to the `node` element naming the object to delete.**

3 **Delete all text between and including the `node` element for that object.**

Using the sample below, to delete the Phone object, delete all the text in the sample.

```
<node-Phone display-order="3" merge-must-delete="false">
   <field-PhoneType>
      <display-name>Phone Type</display-name>
      <display-order>1</display-order>
      <max-length>8</max-length>
      <gui-type>MenuList</gui-type>
      <value-list>PHONTYPE</value-list>
      <value-type>string</value-type>
      <key-type>true</key-type>
   </field-PhoneType>
   <field-Phone>
      <display-name>Phone</display-name>
      <display-order>2</display-order>
      <max-length>20</max-length>
      <gui-type>TextBox</gui-type>
      <value-type>string</value-type>
      <input-mask>(DDD)DDD-DDDD</input-mask>
      <value-mask>xDDDxDDDxDDDD</value-mask>
   </field>
</node>
```

4 **If necessary, renumber the order of the remaining objects so they are sequential.**

5 **Remove the object from the relationship definition, as described in "Defining Relationships Between Master Index Objects (Repository)" on page 29.**

6 **Save and close the file.**

Adding Fields to the EDM

You can define new fields for an object in the Enterprise Data Manager file, but the field must correspond with a field defined for that object in the Object Definition file. Only the fields defined in the Enterprise Data Manager file appear on the EDM windows. If you add a field to the object structure using the Configuration Editor, it is automatically added to the Enterprise Data Manager file.

▼ To Define New Fields

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `node` element that defines the object to which you want to add a field.**

3 **In the `node` element, create a new element named `field-`, where *field_name* is the name of the field as it appears in the Object Definition file.**

For example:

```
<node-Person>
   <field-LastName>
   </field-LastName>
</node>
```

4 **Define and configure the elements listed in"EDM Field Configuration Elements" on page 129 for the new `field-`*field_name* element.**

For example:

```
<field-LastName>
   <display-name>LastName</display-name>
   <display-order>2</display-order>
   <max-length>40</max-length>
   <gui-type>TextBox</gui-type>
   <value-type>string</value-type>
   <is-sensitive>false</is-sensitive>
   <key-type>false</key-type>
</field-LastName>
```

5 **If necessary, renumber any existing fields to keep the numbering sequential.**

6 **Save and close the file.**

EDM Field Configuration Elements

The following table lists and describes the XML elements that define the fields that appear on the Enterprise Data Manager.

Element	Description
name	The name of the field as it appears in the object definition.
display-name	The name of the field as it will appear on the EDM.
display-order	The order in which the field appears in the object on the EDM. For example, specify **1** to indicate this is the first field on the EDM pages, **2** to indicate it is the second field, and so on. The display order goes from left to right,
max-length	The maximum number of characters displayed on the EDM for the field.
gui-type	An indicator of the type of display for the field. Specify **TextBox** for a standard data entry field; **MenuList** for a field that must be populated by selecting from a drop-down list; or **TextArea** for a long field that requires a scrollbar, such as a comments field.

Element	Description
value-type	The Oracle Java CAPS Master Index data type for the data populated in the field. Enter one of the following values: ■ **string** - Field contains a string of characters. ■ **date** - Field contains a date value. ■ **float** - Field contains a floating point integer. ■ **int** - Field contains an integer. ■ **char** - Field contains a single character. ■ **boolean** - Field contains either "true" or "false".
input-mask	A mask used by the EDM to add punctuation to a field. You can add an input mask to display telephone numbers as "(123)456-7890" even though no punctuation is entered by the user. To define an input mask, type a character type for each character in the field, and place any necessary punctuation between the character types. For example, the input mask for the above telephone format is "(DDD)DDD-DDDD". The following character types are allowed: ■ **D** - indicates a numeric character. ■ **L** - indicates an alphabetic character. ■ **A** - indicates an alphanumeric character. **Note –** If the length of the input mask is greater than the value specified for the max-length element, the length of the input mask is used.
value-mask	A mask used by the master index application to strip any extra characters that were added by the input mask. This mask ensures that data is stored in the database in the correct format. This mask must be the same length as the input mask. To specify a value mask, type the same value as is entered for the input mask, but type an "x" in place of each punctuation mark. For example, if an SSN field has an input mask of DDD-DD-DDDD, you need to specify a value mask of DDDxDDxDDDD to strip the dashes before storing the SSN. A value mask is not required for date fields.
value-list	The name of the menu list used to populate the drop-down list for the field. This is required if the `gui-type` specified for the field is MenuList, and it must match a code of an element in the sbyn_common_header database table.
is-sensitive	An indicator of whether the value of the field is hidden on the EDM for records with a VIP status of "VIP". Only users with the required security permissions can view the hidden information. Specify **`true`** to hide the field value; specify **`false`** (or remove the `is-sensitive` element) to display the field value. **Note –** This element is only used if the `object-sensitive-plug-in-class` in the `impl-details` section is populated.

Element	Description
key-type	An indicator of whether the field (or a combination of key fields) must be unique in an enterprise record. Unique key fields identify unique child objects in an enterprise object. For example, if the object structure contains a child object for addresses and each record can have only one of each type of address (for example, one home address and one work address), then the address type field would be a key field. Specify **`true`** to indicate the field is a key field; specify **`false`** if it is not.

Removing Fields From the EDM

If a field is defined for an object in the Enterprise Data Manager file, that field appears on the EDM windows. If there are any fields defined in the object structure that you do not want to display on the EDM, you can remove the field definition from the Enterprise Data Manager file. If you remove a field from the object structure using the Configuration Editor, it is automatically removed from the Enterprise Data Manager file.

Note – If you remove a unique key type field, you must write a custom plug-in that will automatically populate a value into the field in order to meet unique key type field constraints.

▼ To Remove a Field From the EDM

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `field` element that defines the field you want to remove.**

3. **Delete the field elements that define the field you want to remove.**

4. **If necessary, renumber the remaining fields, as described in "Modifying EDM Field Display Options" on page 131.**

5. **Save and close the file.**

Modifying EDM Field Display Options

Once a field is defined for an object in the Enterprise Data Manager file, you can change the name that appears on the EDM for that field, the location of the field, and the type for the field (such as text box, menu list, and so on).

▼ To Modify a Field's Display Options

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `node` element that defines the object that contains the field you want to modify.**

3 **Scroll to the `field` element you want to modify, .**

4 **To modify the field label, change the value of the `display-name` element.**

For example:

```
<display-name>Last Name</display-name>
```

5 **To modify the location of the field on the EDM, change the value of the `display-order` element.**

For example:

```
<display-order>1</display-order>
```

Note – If you change the order of one field, you must change the order of at least one other field to maintain sequential numbering. For example, if you change a field's location from "2" to "1", you must then change the location of the field originally specified for location 1.

---

6 **To modify the type of field to display, change the value of the `gui-type` element.**

For example:

```
<gui-type>TextBox</gui-type>
```

Note – You can enter TextBox (for a standard field), MenuList (for a field whose value must be selected from a drop-down list), or TextArea (for a long field that requires a scrollbar).

---

7 **Save and close the file.**

Specifying a Drop-Down List for an EDM Field

Once a field is defined for an object in the Enterprise Data Manager file, you can specify or change the name of the drop-down list for the field. If you modify the Code Module for the field in the Configuration Editor, the drop–down list is automatically updated in the Enterprise Data Manager file.

▼ To Specify a Drop-Down List

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `node` element that defines the object containing the field you want to modify.**

3. **Scroll to the `field` element you want to modify.**

4. **Enter the name of the drop-down list in the `value-list` element.**

   If the element does not exist for the field, create a new `value-list` element. For example:

   ```
   <value-list>STATE</value-list>
   ```

   Note – The value of the `gui-type` element for the field must be "MenuList" if you specify a drop-down list. The `value-list` element must match a code column value in the sbyn_common_header database table unless the drop-down list is populated by information in the sbyn_user_code table (as they might be for auxiliary IDs). In this case, the `value-list` element must match a code_list column value in sbyn_user_code.

5. **Save and close the file.**

Specifying an EDM Field's Length and Format

Once a field is defined for an object in the Enterprise Data Manager file, you can change the number of characters that can be entered for the field in the EDM. You can also specify whether to automatically enter punctuation into a field on the EDM, but remove the punctuation in the database. If you modify the field's length and format in the object structure using the Configuration Editor, the Enterprise Data Manager file is automatically updated as well.

Note – Field length here is constrained by the length of the database column containing the field and the length defined in the Object Definition file.

▼ To Modify a Field's Length and Format

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `node` element that defines the object containing the field you want to modify.**

3. **Scroll to the `field` element you want to modify.**

4. **To modify the length of a field, change the value of the `max-length` element.**

   For example:

   ```
   <max-length>100</max-length>
   ```

5. **To modify the format of the field, change the value of the `input-mask` and `value-mask` elements.**

   If these element do not exist for the field, create new `input-mask` and `value-mask` elements. For example:

   ```
   <input-mask>(DDD)DDD-DDDD</input-mask>
   <value-mask>xDDDxDDDxDDDD</value-mask>
   ```

   Note – If an input mask is defined, in most cases a value mask must also be defined. For information about input and value masks, see "EDM Field Configuration Elements" on page 129.

6. **Save and close the file.**

Modifying an EDM Field's Data Type

Each field on the EDM requires a specific type of data to be entered. For example, name fields generally require a data string and date fields require a valid date or numeric characters. The type of data defined for each field must correspond with the field type defined for that field in the Object Definition file and in the database. If you modify a field's data type in the object structure using the Configuration Editor, the Enterprise Data Manager file is automatically updated as well.

▼ To Modify the Data Type

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `node` element that defines the object that contains the field you want to modify.**

3. **Scroll to the `field` element you want to modify.**

4. **Change the value of the `value-type` element.**

   For example:

   ```
   <value-type>string</value-type>
   ```

5 **Save and close the file.**

Defining Key Fields for an Object (Repository)

You can specify that a certain field or combination of fields be unique in a system object or SBR. An example of a unique fields would be the address type if only one address of each type is allowed. A field's key type status in the Enterprise Data Manager file must match its key type status in the Object Definition file. If you modify a field's key type status in the object structure using the Configuration Editor, the Enterprise Data Manager file is automatically updated as well.

▼ To Modify the Key Status

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `node` element that defines the object containing the field you want to modify.**

3 **Scroll to the `field` element you want to modify.**

4 **To specify that a field must be unique in a system object, change the value of the `key-type` element to `true`.**

5 **To specify that a field does not need to be unique in a system object, change the value of the `key-type` element to `false`.**

For example:

```
<key-type>false</key-type>
```

6 **To specify that a combination of fields must be unique for an object rather than just one single field, set the `key-type` element to `true` for each field.**

7 **Save and close the file.**

Masking Field Values on the EDM

You can specify that the values of certain fields be hidden on the EDM from users without the appropriate access permissions. You can create a custom plug-in that uses the value of the VIP Flag field to determine whether the values of specified fields are hidden (typically in records with a VIP status of "VIP" or "Employee"). In records with any other VIP status, the field values would be visible regardless of whether the field is marked for masking.

This option is only available if the `object-sensitive-plug-in-class` element in the `impl-details` section is populated with the custom plug-in class (by default, this element is empty).

Note – To mask field values, you need to define a custom plug-in to implement the masking rules.

▼ To Mask Field Values on the EDM

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `node` element that defines the object containing the field you want to modify.**

3. **Scroll to the `field` element you want to modify.**

4. **Change the value of the `is-sensitive` element to `true`.**

   If the element does not exist for the field, create a new `is-sensitive` element. For example:

   ```
   <is-sensitive>true</is-sensitive>
   ```

   Note – The `is-sensitive` element must appear before the `key-type` element in the field definition.

   ---

5. **Save and close the file.**

Defining EDM Object Relationships (Repository)

The relationships in the Enterprise Data Manager file are predefined based on the information you provided when you created the object structure definition (the Object Definition file). The relationship structure in the Enterprise Data Manager file should match that of the Object Definition file.

▼ To Define Relationships

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `relationships` element.**

3 **Specify the name of the parent object in the `name` element.**

4 **Specify the name of the child objects in the `children` elements.**

For example:

```
<relationships>
   <name>Person</name>
   <children>Alias</children>
   <children>Address</children>
   <children>Phone</children>
   <children>AuxId</children>
</relationships>
```

5 **To remove a child object from the relationships list, delete the `children` element defining the object you want to delete.**

6 **Save and close the file.**

Defining EDM Local ID Labels (Repository)

The `system-display-name-overrides` element, which is nested in the `gui-definition` element, defines alternate names for the local ID headings and labels. The name defined here replaces the default local ID heading and field names on all pages, including the search result column names. This element is optional, and if it does not exist the label names default to "Local ID". If the value of either of the sub-elements is missing, the label names also default to "Local ID".

▼ To Define Local ID Labels

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `system-display-name-overrides` element in the `gui-definition` element.**

3 **Modify any of the following elements.**

- `local-id-header` – The name to use for the header label of the local ID search section of the EDM Lookup window.
- `local-id` – The name to use for all fields and columns containing local IDs. In fields where the local ID label is abbreviated to "LID1" or "LID2", the name becomes *local-id*1 or *local-id*2 (where *local-id* is the value specified for this element).

For example:

```
<gui-definition>
   <system-display-name-overrides>
      <local-id-header>System Identifier</local-id-header>
      <local-id>System ID</local-id>
   </system-display-name-overrides>
   ...
```

4 **Save and close the file.**

Configuring the Master Index EDM Search Pages (Repository)

The search pages that appear on the EDM are configured in the Enterprise Data Manager file. The `eo-search` element, which is nested in the `page-definition` element of the `gui-definition` element, contains all of the configuration information for the search pages that appear on the EDM. If you add a new query to the Candidate Select file and you want to access that query from the EDM, you must create a new search page for the query.

Perform any of the following actions to configure the search pages of the EDM. Though some search options can be configured using the Configuration Editor, it is best to modify the XML file directly.

- "Specifying Standard EDM Search Page Properties (Repository)" on page 138
- "Creating a Search Page for the EDM (Repository)" on page 139
- "Modifying a Search Page on the EDM (Repository)" on page 144

Specifying Standard EDM Search Page Properties (Repository)

Standard search properties include the type of object returned by each search, the name of the tabbed header for the search pages, and the URL for entry into the search area. These properties apply to all search pages you define, and they can be modified as needed.

▼ To Specify Standard Search Page Properties

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `page-definition` element in the `gui-definition` element.**

3 **In the `eo-search` element, modify any of the following elements. Do not modify the `tab-entrance` element.**

- `root-object` – The name of the object returned by the search (this must be the name of the parent object).
- `tab-name` – A name for the search pages. This name appears on tab label associated with the search pages on the EDM.

For example:

```
<root-object>Person</root-object>
<tab-name>Customer Search</tab-name>
<tab-entrance>/stcedm/EnterEOSearchSimpleAction.do</tab-entrance>
```

4 **Save and close the file.**

Creating a Search Page for the EDM (Repository)

Several search pages are defined by default by the wizard, including the simple lookup page, advanced lookup pages, and the comparison lookup page. You can create additional search pages if needed. Perform the following steps to create a new search page.

- "Step 1: Define the Search Page" on page 139
- "Step 2: Define the Search Fields" on page 140
- "Step 3: Specify Search Options" on page 141

Step 1: Define the Search Page

The first step in creating a search page is to define certain properties for the appearance of the page, such as its name, how many fields to list in each row, whether to display the EUID or local ID field, and general instructions for the search.

Note – If either the EUID field or the local ID and system fields appear on a search page, any values entered into these optional fields take precedence over information entered into other search fields. For example, if an invalid EUID is entered but valid first and last names are entered, no results are returned due to the invalid EUID. The EUID field takes precedence over the local ID and system fields.

▼ To Define the Search Page

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `page-definition` element in the `gui-definition` element.**

3 **In the `eo-search` element, create a `simple-search-page` element.**

Make sure the new element falls within the `eo-search` element, but outside any existing `simple-search-page` elements. For example:

```
<eo-search>
   <simple-search-page>
    ...
   </simple-search-page>
   <simple-search-page>
   </simple-search-page>
</eo-search>
```

4 **In the new `simple-search-page` element, create the elements listed in "EDM Search Page Definition Elements" on page 142 and enter the appropriate value for each element.**

For example:

```
<eo-search>
   <simple-search-page>
    ...
   </simple-search-page>
   <simple-search-page>
      <screen-title>Address Search</screen-title>
      <field-per-row>1</field-per-row>
      <show-euid>true</show-euid>
      <show-lid>false</show-lid>
      <instruction>Enter address information below.</instruction>
   </simple-search-page>
</eo-search>
```

5 **Continue to "Step 2: Define the Search Fields" on page 140.**

Step 2: Define the Search Fields

Once you define the search page, you must specify the fields that appear on the page. Fields are specified in field groups, and each field group represents a boxed area on the search page. All fields specified for a field group appear in the boxed area named by that group. The box label is defined by the description of the field group.

▼ To Define Search Fields

1 **Complete "Step 1: Define the Search Page" on page 139.**

2 **In the new `simple-search-page` element, create a `field-group` element.**

For example:

```
<simple-search-page>
   <screen-title>Simple Person Search</screen-title>
   <field-per-row>2</field-per-row>
   <show-euid>false</show-euid>
   <show-lid>false</show-lid>
   <field-group>
   </field-group>
</simple-search-page>
```

3 **In the new `field-group` element, create the elements and attributes listed in "EDM Search Field Definition Elements" on page 142 and enter the appropriate value for each.**

For example:

```
<field-group>
   <description>Address</description>
   <field-ref>Address.AddressType</field-ref>
   <field-ref>Address.AddressLine1</field-ref>
   <field-ref>Address.AddressLine2</field-ref>
   <field-ref required="true">Address.City</field-ref>
   <field-ref>Address.State</field-ref>
</field-group>
```

4 **Repeat steps 2 and 3 for each field group you want to display on the selected search page.**

5 **Continue to "Step 3: Specify Search Options" on page 141.**

Step 3: Specify Search Options

After you define the criteria fields for the EDM search, you must specify certain options for the search, such as the types of available searches, whether each search is weighted, and whether the search allows wildcard characters.

▼ To Specify Search Options

1 **Complete "Step 2: Define the Search Fields" on page 140.**

2 **In the `simple-search-page` element you created, create a `search-option` element.**

For example:

```
<simple-search-page>
   <screen-title>Simple Person Search</screen-title>
   <field-per-row>2</field-per-row>
   <show-euid>false</show-euid>
   <show-lid>false</show-lid>
   <field-group>
    ...
   </field-group>
   <search-option>
   </search-option>
</simple-search-page>
```

3 **In the new `search-option` element, create the elements listed in "EDM Search Option Elements" on page 144 and enter the appropriate value for each element.**

For example:

```
<search-option>
   <display-name>Alpha Search</display-name>
   <query-builder>ALPHA-SEARCH</query-builder>
   <weighted>false</weighted>
   <candidate-threshold>2000</candidate-threshold>
```

```
   <parameter>
     <name>UseWildCard</name>
     <value>true</value>
   </parameter>
</search-option>
```

4 **Repeat the previous two steps for each search type you want to make available on the selected search page.**

Note – If you define multiple search option elements, an option button (labelled by the value of the `display-name` element) appears on the search page for each search option.

---

5 **Save and close the file.**

EDM Search Page Definition Elements

The following table lists and describes the elements you can configure in the Enterprise Data Manager file to define the search pages on the EDM.

Element	Description
screen-title	The name of the search as it appears in the Search Type drop-down list, from which users can select a type of search to perform.
field-per-row	The number of fields to display in each row on the search page.
show-euid	An indicator of whether to display the EUID. Specify **true** to display the EUID; otherwise specify **false**. Only display this field if you want it to take precedence over all other search criteria. When the EUID is displayed, it appears in its own labelled box.
show-lid	An indicator of whether to display the local ID and system fields. Specify **true** to display the fields; otherwise specify **false**. Only display these fields if you want them to take precedence over all other search criteria (except the EUID field). When the local ID is displayed, the local ID and system fields appear in their own labelled box.
instruction	A short statement to help the user process a search. The text you enter here appears above the search fields on the Search page.

EDM Search Field Definition Elements

The following table lists and describes the elements you can configure in the Enterprise Data Manager file to define the fields for each search on the EDM.

Element/Attribute	Description
description	A description of the fields defined for the `field-group` element. This value appears as a box label for the area of the page that contains the specified fields.
field-ref	The simple field names of the fields in the field group using their corresponding objects as the root. For example, the path to the FirstName field in the Person object is "Person.FirstName". You can define multiple `field-ref` elements for each field group.
field-ref/required	An indicator of whether the field is required in order to perform a search. Specify any of the following values: ■ **true** – The corresponding field is required to perform the search. These fields are marked with an asterisk (*) on the search page. ■ **false** – The corresponding field is not required to perform the search. If the `required` attribute is not defined, the default is false. ■ **one of** – At least one of the fields with this designation is required to perform the search. This is used to specify that at least one field in the group of fields with the "one of" designation is required. If a group of fields is designated as "one of", those fields are marked with a dagger (†) on the search page. **Tip –** If you make a field required for a search, it is a good idea to make it required when creating a record as well (by specifying true for the required property for the field in the Object Definition file. Otherwise, searches performed from the EDM could result in no possible matches even though possible matches exist.
field-ref/choice	An indicator of whether the field allows you to search by a range of values rather than an exact value. Specify any of the following values: ■ **exact** – The search is performed on the exact value entered (wildcard characters may be allowed). If the `choice` attribute is not specified, this is the default value. ■ **range** – The search is performed on a range of values based on the entered search criteria. Fields with this designation appear twice on the search page, once with "From" appended to the field label and once with "to" appended to the field label. If you specify "range" for a field in a search that uses a blocking query, be sure to modify the query block in the Candidate Select file accordingly. **Tip –** You can specify the same field for both exact and range searching by adding it twice to the field list with different attribute values, giving the choice of performing an exact search or a range search from the EDM. For more information about range searching, see "Candidate Select Configuration (Repository)" in *Understanding Oracle Java CAPS Master Index Configuration Options (Repository)*.

EDM Search Option Elements

The following table lists and describes the elements you can configure in the Enterprise Data Manager file to define the attributes for each search on the EDM, such as which query to use, whether the search results are weighted, and so on.

Element	Description
display-name	A short phrase describing the type of search to perform, such as "Alphanumeric Search" or "Phonetic Search". This appears next to the option button on the search page when multiple search options are defined.
query-builder	The type of query to use when this type of search is selected. The value entered here must match a `query-builder` name in the Candidate Select file.
weighted	An indicator of whether the results of the search are assigned matching probability weights. Specify **true** to assign matching weights; specify **false** to return unweighted results.
candidate-threshold	The maximum number of records to return for a search. This value must be a positive number, and is only used for blocking queries. Setting the candidate threshold to zero is equivalent to not setting a threshold.
parameter	A list of optional parameters for the search, specified by `name` and `value` elements (described below).
name	The name of the parameter. Currently, only `UseWildCard` is available.
value	The value of the parameter. For the `UseWildCard` parameter, this is an indicator of whether the parameter is enabled or disabled. Specify **true** to allow wildcard characters; specify **false** to perform exact-match searches.

Modifying a Search Page on the EDM (Repository)

Once a search page is defined, it can be modified as needed. You can perform any of the following actions to customize existing search page elements.

- "Modifying a Search Page Definition" on page 144
- "Modifying Search Fields" on page 145
- "Modifying Search Options" on page 146

Modifying a Search Page Definition

Once a search page is defined in the Enterprise Data Manager file, you can modify the search page definition. The following properties can be modified: the name of the search, the number of fields that appear on each row of the search page, and whether the EUID or local ID fields are visible.

▼ To Modify a Search Page Definition

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `page-definition` element in the `gui-definition` element.**

3. **Scroll to the `eo-search` element, and then to the `simple-search-page` element you want to modify.**

4. **In the `simple-search-page` element, change the value of any of the elements listed in "EDM Search Page Definition Elements" on page 142.**

   For example:

   ```
   <simple-search-page>
      <screen-title>Customer Search</screen-title>
      <field-per-row>2</field-per-row>
      <show-euid>true</show-euid>
      <show-lid>false</show-lid>
      <instruction>Enter the EUID below.</instruction>
   </simple-search-page>
   ```

5. **Save and close the file.**

Modifying Search Fields

Once field groups and fields are defined for a search page, you can modify the properties of the group and of the fields contained in a group. For more information about the elements that contain the search field configuration, see "EDM Search Field Definition Elements" on page 142.

▼ To Modify Search Fields

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `page-definition` element in the `gui-definition` element.**

3. **Scroll to the `eo-search` element, and then to the `simple-search-page` element you want to modify.**

4 **In the `simple-search-page` element, scroll to the `field-group` you want to modify, and do any of the following:**

- **To modify the name of the boxed area in which the field group appears in the EDM, change the value of the `description` element.**
- **To add a new field to a field group, create and name a new `field-ref` element in the appropriate `field-group` element.**
- **To modify the name of a field defined for a field group, change the value of the appropriate `field-ref` element.**
- **To specify whether a field is required, add a `required` attribute and specify a value defined in "EDM Search Field Definition Elements" on page 142.**
- **To specify whether a field is used for range searching, add a `choice` attribute and specify a value defined in "EDM Search Field Definition Elements" on page 142.**
- **To delete a field from a field group, delete all text between and including the `field-ref` tags that define the field to be deleted.**
- **To delete an entire field group, delete all text between and including the `field-group` tags that define the field group to be deleted.**

5 **Save and close the file.**

Modifying Search Options

Once search options are defined for a search page, you can modify those options if needed. For more information about the elements that define search options, see "EDM Search Option Elements" on page 144.

▼ To Modify Search Options

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `page-definition` element in the `gui-definition` element.**

3 **Scroll to the `eo-search` element, and then to the `simple-search-page` element you want to modify.**

4. **In the new `simple-search-page` element, scroll to the `search-option` element and do any of the following:**

   - **To modify the name of the search option button, change the value of the `display-name` element.**
   - **To modify the query type of the selected search, change the value of the `query-builder` element. The query you specify must match a query defined in the Candidate Select file.**
   - **To specify that a search return weighted results, change the value of the `weighted` element to true.**
   - **To specify that a search return unweighted results, change the value of the `weighted` element to false.**
   - **To specify that wildcard characters can be used in a search, change the `UseWildCard` parameter `value` element to true.**
   - **To specify that wildcard characters cannot be used in a search, change the `UseWildCard` parameter `value` element to false.**

5. **Save and close the file.**

Configuring Master Index EDM Page Layouts (Repository)

You can define how fields appear on most EDM pages in the Enterprise Data Manager file. You can also define the name of the tabbed heading for a page, the type of object handled on each page, and the first page to appear when a user logs on.

Perform any of the following actions to define screen layouts. Modify the XML file directly to perform these tasks.

Specifying the Initial View for the EDM (Repository)

By default, the Matching Review page appears when a user logs on to the Enterprise Data Manager. You can specify any of the other tabbed pages as the initial view.

▼ To Specify the Initial View

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `page-definition` element in the `gui-definition` element, and then to the `initial-screen` element.**

3. **Do one of the following:**

   - **To display the Matching Review page first, change the value of the `initial-screen` element to "Matching Review".**
   - **To display the Search page first, change the value of the `initial-screen` element to "EO Search".**
   - **To display the Create System Record page first, change the value of the initial-screen element to "Create System Record".**
   - **To display the History page first, change the value of the `initial-screen` element to "History".**
   - **To display the Reports page first, change the value of the `initial-screen` element to "Reports".**

   Note – These values remain the same regardless of whether the tab name of the page has been changed. For example, if you change the name of the Matching Review page to "Potential Duplicate", you would still specify "Matching Review" for that page to appear first.

4. **Save and close the file.**

Configuring the EDM Search Results Page (Repository)

The Search Results page displays a list of records returned from a search. The same Search Results page appears for all simple search pages. You can configure the number of records to

display at one time in the results list and the fields to display in the list. Fields in the `field-ref` elements are named by their object name and then the field name; for example, Person.LastName or Phone.PhoneNumber.

▼ To Configure the Search Results Page

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `page-definition` element in the `gui-definition` element, and then to the `eo-search` element.**

3. **In the `search-result-list-page` element, do any of the following:**

   - **To modify the number of records displayed on the search results page at one time, change the value of the `item-per-page` element.**

     For example:

     ```
     <item-per-page>15</item-per-page>
     ```

   - **To specify a maximum number of records to return from a search, change the value of the `max-result-size` element.**

     For example:

     ```
     <max-result-size>100</max-result-size>
     ```

   - **To add a new field to the results list, create and name a new `field-ref` element.**

     For example:

     ```
     <field-ref>Person.LastName</field-ref>
     ```

   - **To modify a field in the results list, change the value of the appropriate `field-ref` element.**

     For example:

     ```
     <field-ref>Person.FirstName</field-ref>
     ```

   - **To delete a field from the results list, delete all text between and including the `field-ref` tags defining the field to be deleted.**

     For example, to delete the City field from the following list, delete the boldface text.

     ```
     <search-result-list-page>
        <item-per-page>10</item-per-page>
        <field-ref>Person.LastName</field-ref>
        <field-ref>Person.FirstName</field-ref>
        <field-ref>Address.AddressLine1</field-ref>
        <field-ref>Address.City</field-ref>
     </search-result-list-page>
     ```

4 **Save and close the file.**

Configuring the EDM View/Edit Page (Repository)

The View/Edit page appears when you select a record in the search results list. This page displays all of the information about an enterprise object. You can configure the number of fields that appear in each row of the view page.

▼ To Configure the View/Edit Page

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `page-definition` element in the `gui-definition` element, and then to the `eo-search` element.**

3 **In the `eo-view-page` element, change the value of the `field-per-row` element.**

For example:

```
<eo-view-page>
   <field-per-row>2</field-per-row>
</eo-view-page>
```

4 **Save and close the file.**

Configuring the EDM Create System Record Page (Repository)

The Create System Record page is where a user adds new records to the master index database. You can configure the name of the tabbed heading and the type of object returned. Do not modify the URL of the entrance to the Create System Record page. Following is a sample of the `create-eo` element.

```
<create-eo>
   <root-object>Person</root-object>
   <tab-name>Create System Record</tab-name>
   <tab-entrance>/stcedm/EnterEOCreateAction.do</tab-entrance>
</create-eo>
```

▼ To Configure the Create EO Page

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `page-definition` element in the `gui-definition` element, and then to the `create-eo` element.**

3. **To specify a new object type for the objects you create, change the value of the `root-object` element.**

   For example:

   ```
   <root-object>Customer</root-object>
   ```

   Note – This must be the name of the parent object.

4. **To specify a new name for the Create System Record tab, modify the value of the *tab-name* element.**

   ```
   <tab-name>Add Record</tab-name>
   ```

5. **Save and close the file.**

Configuring the EDM History Page (Repository)

The History page allows you to search for records, view a results list, and then view a history of changes to the record you select. You can configure the name of the tabbed heading for the page, the type of object returned, the appearance of the search and results pages, and the appearance of the merge history page. Do not modify the URL of the entrance to the History page. Following is a sample of the `history` element.

```
<history>
   <root-object>Company</root-object>
   <tab-name>History</tab-name>
   <tab-entrance>/stcedm/EnterXASearchAction.do</tab-entrance>
   <xa-search-page>
      <field-per-row>2</field-per-row>
   </xa-search-page>
   <search-result-list-page>
      <item-per-page>10</item-per-page>
      <max-result-size>200</max-result-size>
      <field-ref>Person.FirstName</field-ref>
      <field-ref>Person.LastName</field-ref>
      <field-ref>Phone.Phone</field-ref>
      <field-ref>Address.AddressLine1</field-ref>
   </search-result-list-page>
```

```
   <merge-history-key-field>
      <field-ref>Person.FirstName</field-ref>
      <field-ref>Person.LastName</field-ref>
   </merge-history-key-field>
</history>
```

▼ To Configure the History Page

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `page-definition` element in the `gui-definition` element, and then to the `history` element.**

3 **To specify a new object to be returned from a history search, change the value of the `root-object` element.**

For example:

```
<root-object>Customer</root-object>
```

Note – This must be the name of the parent object.

---

4 **To specify a new name for the History tab, modify the value of the `tab-name` element.**

For example:

```
<tab-name>Transaction History</tab-name>
```

5 **To specify the number of fields to display in each row on the History search page, change the value of the `field-per-row` element in the `xa-search-page` element.**

For example:

```
<xa-search-page>
   <field-per-row>3</field-per-row>
</xa-search-page>
```

6 **To specify the number of records to display on the History search results page, change the value of the `item-per-page` element in the `search-result-list-page` element.**

For example:

```
<item-per-page>15</item-per-page>
```

7 **To specify a maximum number of records to return from a History search, change the value of the `max-result-size` element in the `search-result-list-page` element.**

For example:

```
<max-result-size>250</max-result-size>
```

8. **To customize the fields that appear in the results list, do any of the following in the `search-result-list-page` element:**
   - **Modify the value of an existing `field-ref` element to the simple field name of a field you want to appear in the results list.**
   - **Add and name a new `field-ref` element (using the simple field name).**
   - **Delete a `field-ref` element defining a field you do not want to appear in the results list.**
9. **To customize the fields that appear on the merge history page, do any of the following in the `merge-history-key-field` element:**
   - **Modify the value of an existing `field-ref` element to the simple field name of a field you want to appear on the merge history page.**
   - **Add and name a new `field-ref` element (using the simple field name).**
   - **Delete a `field-ref` element defining a field you do not want to appear on the merge history page.**
10. **Save and close the file.**

Configuring the EDM Match Review Page (Repository)

The Match Review page allows you to search for potential duplicate or assumed match records to compare, view a results list, and then view a comparison of the records you select. You can configure the name of the tabbed heading for the page, the type of object returned, and the appearance of the search and results pages. Do not modify the URL of the entrance to the Match Review page. Following is a sample of the `matching-review` element.

```
<matching-review>
   <root-object>Company</root-object>
   <tab-name>Matching Review</tab-name>
   <tab-entrance>/stcedm/EnterPDSearchAction.do</tab-entrance>
   <pd-search-page>
      <field-per-row>2</field-per-row>
   </pd-search-page>
   <search-result-list-page>
      <item-per-page>10</item-per-page>
      <max-result-size>250</max-result-size>
   </search-result-list-page>
</matching-review>
```

▼ To Configure the Match Review Page

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `page-definition` element in the `gui-definition` element, and then to the `matching-review` element.**

3 **To specify a different object to be returned from the search, change the value of the `root-object` element.**

For example:

```
<root-object>Customer</root-object>
```

Note – This must be the name of the parent object.

---

4 **To specify a different name for the Matching Review tab, modify the value of the `tab-name` element.**

For example:

```
   <tab-name>Potential Duplicates</tab-name>
```

5 **To specify the number of fields to display in each row on the Matching Review search page, change the value of the `field-per-row` element in the `pd-search-page` element.**

For example:

```
<pd-search-page>
   <field-per-row>3</field-per-row>
</pd-search-page>
```

6 **To specify the number of records to display on the Matching Review search results page, change the value of the `item-per-page` element in the `search-result-list-page` element.**

For example:

```
<item-per-page>15</item-per-page>
```

7 **To specify a maximum number of results for a Matching Review search, change the value of the `max-result-size` element in the `search-result-list-page` element.**

For example:

```
<max-result-size>100</max-result-size>
```

8 **Save and close the file.**

Configuring the EDM Reports and Reports Page (Repository)

Configuring the Reports page consists of two steps: configuring the page itself and configuring the individual reports.

Configuring the Reports Page

For the Reports page, you can configure the name of the tabbed heading for the page and the type of object returned. Do not modify the URL of the entrance to the Reports page. Following is a sample of the Reports page configuration elements.

```
<reports>
   root-object>Company</root-object>
   <tab-name>Reports</tab-name>
   <tab-entrance>/EnterReportSearchAction.do</tab-entrance>
   <search-page-field-per-row>2</search-page-field-per-row>
```

The report configuration section of the Enterprise Data Manager configuration file defines the appearance of the Reports page, and is located within a set of `reports` tags near the end of the file.

▼ To Configure the Reports Page

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `reports` element. This is located near the end of the file.**

3. **Modify the values for any of the elements described in Table 1.**

4. **When you have finished configuring the Reports page, save the file.**

Configuring Reports

A configuration section is defined for each of the six production report templates and for each of the three activity reports. Use these sections to configure each production and activity report to display information as you want to view it. You can also specify which reports can be run from the EDM. Following is an example of a report configuration stanza.

```
<report name="Potential Duplicate" title="Potential Duplicate Report">
   <enable>true</enable>
   <max-result-size>1000</max-result-size>
   <page-size>100</page-seize>
   <fields>
```

```
      <field-ref>Company.CompanyName</field-ref>
      <field-ref>Company.CompanyType</field-ref>
      <field-ref>Company.StockSymbol</field-ref>
      <field-ref>Company.ContactPerson</field-ref>
      <field-ref>Phone.Phone</field-ref>
      <field-ref>Address.AddressLine1</field-ref>
      <field-ref>Address.AddressLine2</field-ref>
   </fields>
</report>
```

▼ To Configure Reports

Perform the following steps for each production and activity report.

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `reports` element. This section is located near the end of the file.**

3. **For each report, specify values for the elements and attributes listed in Table 2.**

4. **For production reports only, define the fields to include on the report by modifying, adding, or removing `field` elements.**

5. **When you have finished configuring each report, save and close the file.**

EDM Reports and Reports Page Configuration Elements (Repository)

The following table lists and describes the XML elements that define the configuration of the Reports page on the EDM.

TABLE 1 Reports Page Configuration Elements

Element	Description
root-object	The name of the type of object on which to report (this must be the parent object).
tab-name	A name for the report pages. This name appears on tab label associated with the report pages on the EDM.
tab-entrance	The URL to the entry page of the reports pages. This element should not be modified.
search-page-field-per-row	The number of fields to display in each row of the Reports Search page.

The following table lists and describes the XML elements that define the reports run from the EDM.

TABLE 2 Reports Configuration Elements

Element	Description
report	A report definition of one type of report run from the EDM. with the exception of search reports (which do not need to be configured).
report/name attribute	The type of report being generated. you should not need to modify this element, but you can specify any of the following production reports: ■ Assumed Match ■ Potential Duplicate ■ Deactivated ■ Merged ■ Unmerged ■ Update
report/title attribute	The descriptive name of the report. This can be any string, and appears as the title on the specified report.
enable	An indicator of whether the report can be run from the EDM. Specify **`true`** to allow the report to be run; specify **`false`** to disable the report.
max-result-size	The number of records to display on the report. if no value is entered, or if the value is zero (0), the size defaults to 1000 records. To retrieve all records for a report, enter a very large value for this element.
page-size	The number of records returned to the report generator at one time for each report. If you do not enter a page size or you enter "0", the size defaults to 500 records for all reports.
fields/field-ref	A list of fields to display on the report in addition to those that are displayed by default. Use the simple field name for the `field-ref` value. This element should be empty for the activity reports. If a list of fields is supplied for any activity reports, it is ignored.

Configuring the EDM Audit Log Pages (Repository)

When enabled, the audit log stores a history of each instance in which information from the object tables in the master index database is accessed. The EDM allows you to search for and view the audit log entries. You can enable or disable the audit log in the Enterprise Data Manager file.

▼ To Configure the Audit Log Pages

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `page-definition` element in the `gui-definition` element, and then to the `audit-log` element.**

3 **To specify that records be written to the audit log, change the value of the `allow-insert` element to true.**

For example:

```
<audit-log>
   <allow-insert>true</allow-insert>
</audit-log>
```

4 **To specify that records not be written to the audit log, change the value of the `allow-insert` element to false.**

For example:

```
<audit-log>
   <allow-insert>false</allow-insert>
</audit-log>
```

5 **Save and close the file.**

Configuring Master Index EDM Implementation Information (Repository)

Certain configuration information is defined automatically in the implementation details section of the Enterprise Data Manager file based on information you specify in the wizard. Other information in this section must be customized by modifying the XML file directly.

You can customize the implementation details by performing any of the following tasks.

- "Specifying the Master Controller JNDI Class (Repository)" on page 159
- "Specifying the Master Index Report Generator JNDI Class (Repository)" on page 159
- "Specifying Master Index Validation Services (Repository)" on page 160
- "Setting Master Index Debug Options (Repository)" on page 160
- "Specifying a Master Index Field Masking Class (Repository)" on page 161

Specifying the Master Controller JNDI Class (Repository)

The EDM must know the name of the Master Controller interface for the Oracle Java CAPS Master Index implementation. Only change this value if you create a custom Master Controller.

▼ To Specify the Master Controller JNDI Class

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `impl-details` element, and then to the `master-controller-jndi-name` element.**

3. **Modify the value of the `master-controller-jndi-name` element to the name of the Master Controller JNDI class for your server.**

   For example:

   ```
   <master-controller-jndi-name>ejb/CustomerMasterController
   </master-controller-jndi-name>
   ```

4. **Save and close the file.**

Specifying the Master Index Report Generator JNDI Class (Repository)

The EDM must know the name of the class used to generate reports for the EDM. Only change this value if you create a custom report generator.

▼ To Specify the Report Generator JNDI Class

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `impl-details` element, and then to the `reportgenerator-jndi-name` element.**

3. **Modify the value of the `reportgenerator-jndi-name` element to the name of the report generator JNDI class to use.**

   For example:

   ```
   reportgenerator-jndi-name>ejb/CustomerReportGenerator
   </reportgenerator-jndi-name>
   ```

4 **Save and close the file.**

Specifying Master Index Validation Services (Repository)

Validation services verify processing codes for data entered into the master index database. Only change the names of the validation services if you created custom code list validators.

▼ To Specify the Validation Service

1 **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

The file opens in the NetBeans XML editor.

2 **Scroll to the `impl-details` element, and then to the `validation-service-jndi-name` element.**

3 **To change the name of the validator for processing codes stored in sbyn_common_header, modify the value of the `validation-service-jndi-name` element to the name of the JNDI class used for validation.**

For example:

```
<validation-service-jndi-name>ejb/CustomerCodeLookup
</validation-service-jndi-name>
```

4 **To change the name of the validator for processing codes stored in sbyn_user_code, modify the value of the `usercode-jndi-name` element to the name of the JNDI class used for validation.**

For example:

```
<usercode-jndi-name>ejb/CustomerUserCodeLookup
</usercode-jndi-name>
```

5 **Save and close the file.**

Setting Master Index Debug Options (Repository)

When you first implement Oracle Java CAPS Master Index, you might want to view detailed debug information until you are certain the application is working as required. You can set debug options in the implementation details.

▼ To Set Debug Options

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `impl-details` element, and then to the `debug-flag` element.**

3. **Modify the value of the `debug-flag` element to indicate whether you want debug information logged.**

   For example:

   ```
   <debug-flag>true</debug-flag>
   ```

   Specify **`true`** to log debug information; specify **`false`** to turn logging off.

4. **Modify the value of the `debug-dest` element to indicate where to print debug information.**

   For example:

   ```
   <debug-dest>console</debug-dest>
   ```

   Specify **`console`** to log debug information to a monitor; specify **`file`** to print to a file.

5. **Save and close the file.**

Specifying a Master Index Field Masking Class (Repository)

If you implement a custom class to define field masking logic, you must specify the new class in the implementation details. The custom class must be created as a custom plug-in in the Oracle Java CAPS Master Index project. This class can be used to mask the value of any field for which the `is-sensitive` element is set to "true".

▼ To Specify a Field Masking Class

1. **In the Projects window, expand the Configuration node in the project you want to modify, and then double-click the Enterprise Data Manager file.**

   The file opens in the NetBeans XML editor.

2. **Scroll to the `impl-details` element.**

3. **Change the value of the `object-sensitive-plug-in-class` element to the name of the custom class.**

   For example:

   ```
   <object-sensitive-plug-in-class>
    com.stc.eindex.user.customfieldmasker
   </object-sensitive-plug-in-class>
   ```

4. **Save and close the file.**