Oracle® Fusion Middleware

Customer Data Services Pack Guide for Enterprise Data Quality

Release 12*c* (12.2.1)

E56636-01

October 2015

Describes how to install, manage, and customize the Oracle Enterprise Data Quality Customer Data Services Pack.

ORACLE

Oracle Fusion Middleware Customer Data Services Pack Guide for Enterprise Data Quality, Release 12*c* (12.2.1)

E56636-01

Primary Author: Oracle Corporation

Contents

2 Customizing Customer Data Services Pack

3 Using Matching

4 Installing and Using Data Quality Health Check

5 Using Business Services

Preface

This document describes how to install, manage, and customize the Oracle Enterprise Data Quality Customer Data Services Pack.

Audience

This document is intended for system administrators or application developers who are installing the Oracle Enterprise Data Quality Customer Data Services Pack. It is assumed that you have a basic understanding of application server and web technology and have a general understanding of Linux, UNIX, and Windows platforms.

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

For more information, see the following documents in the Oracle Enterprise Data Quality documentation set.

EDQ Documentation Library

The following publications are provided to help you install and use EDQ:

- *Oracle Fusion Middleware Release Notes for Enterprise Data Quality*
- *Oracle Fusion Middleware Installing and Configuring Enterprise Data Quality*
- *Oracle Fusion Middleware Administering Enterprise Data Quality*
- *Oracle Fusion Middleware Understanding Enterprise Data Quality*
- *Oracle Fusion Middleware Integrating Enterprise Data Quality With External Systems*
- *Oracle Fusion Middleware Securing Oracle Enterprise Data Quality*

- *Oracle Enterprise Data Quality Address Verification Server Installation and Upgrade Guide*
- *Oracle Enterprise Data Quality Address Verification Server Release Notes*

Find the latest version of these guides and all of the Oracle product documentation at:

https://docs.oracle.com

Online Help

Online help is provided for all Oracle Enterprise Data Quality user applications. It is accessed in each application by pressing the **F1** key or by clicking the Help icons. The main nodes in the Director project browser have integrated links to help pages. To access them, either select a node and then press **F1**, or right-click on an object in the Project Browser and then select **Help**. The EDQ processors in the Director Tool Palette have integrated help topics, as well. To access them, right-click on a processor on the canvas and then select **Processor Help**, or left-click on a processor on the canvas or tool palette and then press **F1**.

Conventions

The following text conventions are used in this document:

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

What's New In This Guide

This section summarizes the new features and significant product changes for Oracle Enterprise Data Quality Customer Data Services Pack (EDQ-CDS) in the Oracle Fusion Middleware Release 12*c* (12.2.1) release.

New and Changed Features for Release 12*c* (12.2.1)

Oracle Fusion Middleware Release 12*c* (12.2.1) contains the following new and changed features for EDQ-CDS:

- Improved Control of Key Method Enablement
- Key Method Analysis
- Output of New Match Metadata
- Matching on Customer-Added Attributes
- Inverted EIDs
- Adjusting Match Attribute Weightings

Improved Control of Key Method Enablement

This feature deprecates the existing CDS cluster level mechanism with a more granular and flexible mechanism based on key profiles which specify the enablement of key methods. This feature extends EDQ-CDS to offer a wider menu of key method algorithms.

For more information, see:

- Matching Properties
- Adjusting Matching
- Using Key Generation
- Key Generation Services

Key Method Analysis

Key method analysis introduces the capability within CDS to automatically analyze the customer's data and determine the best key profile for that particular data set.

For more information, see Key Method Analysis.

Output of New Match Metadata

The output of match metadata provides granular details about why two records matched, and which logical match identifiers contributed to a match.

For more information, see Output of Match Metadata.

Matching on Customer-Added Attributes

Matching with customer-added string and date attributes improves how you can configure EDQ and reduces the need to customize the EDQ-CDS configuration for attributes not present on the standard interface

For more information, see:

- Using Matching with Customer-Added Attributes
- Keys for Custom Attributes

Inverted EIDs

Inverted ID matching provides similar functionality to Elimination Ids (EIDs) but produces a "No match" result when the identifier values are the same. Inverted ID matching allows you to eliminate matches where records share a common value.

For more information, see Using Inverted Elimination ID Matching.

Adjusting Match Attribute Weightings

For individuals and entities, all previous static rules (other than UID and elimination rules) have been deleted and replaced by a single 'Overall score' rule which takes the aggregate score from various new compound comparisons with their associated weightings. The original base comparisons are still used as the basis for the compound comparison rule fragments.

The new compound comparisons together with their default weightings are listed for information purposes in the new 'Match - Compound Comparison Default Weightings' reference data inside the EDQ-CDS project.

1

Installing Customer Data Services Pack

This chapter explains how to install the EDQ-CDS.

This chapter includes the following sections:

1.1 Planning Your Installation

This section describes the prerequisites, integration, compatibility, and necessary installation components.

1.1.1 Prerequisites

EDQ-CDS Release 12*c* (12.2.1) requires the following:

- If you are integrating EDQ-CDS with Siebel, you must install:
  - Siebel CRM or UCM version 8.1 or later.
  - Siebel Connector release 12.2.1.

The requirements for production systems are as follows:

- 64-bit Operating System.
- 64-bit Java Virtual Machine (JVM).
- Minimum system memory of 8GB, with 4GB allocated to the JVM.
- Recommended system memory of 16GB, with 8GB allocated to the JVM.

> **Note :** It may be possible to run Test or Development instances on 32-bit systems with less memory.

1.1.2 Integrating with Siebel

When integrating a Siebel instance with EDQ to use CDS services, Oracle recommends that the necessary components be installed and configured in the following order:

1. Install the EDQ-CDS pack on the EDQ server as detailed in this chapter.

2. The `siebelconnector.zip` is automatically installed into the EDQ Oracle Home when the EDQ installer is run. Copy the zip to the Siebel server installation directory.
3. Install the EDQ Siebel Connector on the Siebel server.
4. Integrate Siebel with EDQ-CDS, see *Oracle Fusion Middleware Integrating and Managing Siebel Environments with Enterprise Data Quality.*

1.1.3 Compatibility Matrix

The matrix below shows the compatibility of all released versions of EDQ-CDS with other EDQ components:

EDQ-CDS	EDQ	EDQ Siebel Connector	EDQ-AV
9.0.1	9.0.3 or later	9.0.3-9.0.5	Any
9.0.2	9.0.4 or later	9.0.4-9.0.5	Any
9.0.3	9.0.5 or later	9.0.4-9.0.5	Any
9.0.4	9.0.7 or later	9.0.6	12.4.0.0.0 or later
9.0.5	9.0.7 or later	9.0.6	12.4.0.0.0 or later
11.1.1.7.3	11.1.1.7.3 or later	11.1.1.7.3	12.4.0.0.0 or later
12.2.1	12.2.1 or later	12.2.1	15.2.0.0.0

1.1.4 Components

CDS is delivered with EDQ. To access the CDS `.dxi` files, right-click the server name in EDQ and select **Open Server Package File > cds** folder. The CDS folder contains the following components:

- `EDQ-CDS.dxi` - the EDQ-CDS data quality services.
- `EDQ-CDS - Published Processors.dxi` - contains the Published Processors and Images.
- `EDQ-CDS - Initialize Reference Data.dxi` - the processes to prepare the EDQ-CDS Reference Data.
- `EDQ-CDS - Data Quality Health Check.dxi` - the processes for the Data Quality Health Check extension, see Chapter 4, "Installing and Using Data Quality Health Check."
- `Staging_tables.sql` in `Oracle_Home/edq/oracle.edq/edq-cds` contains Siebel specific scripts for configuring the staging database and a default Structured Query Language (SQL) script for use in creating staging tables for use with generic batch jobs.
- `siebelconnector.zip` contains the `dnd.properties` file, which is used when EDQ-CDS is integrated with a Siebel server. For more information, see *Oracle Fusion Middleware Integrating and Configuring Siebel Environments with Enterprise Data Quality*.

1.2 Installing EDQ-CDS

To install EDQ-CDS on the EDQ server:

1. Start the **EDQ Director client**, and log on as a user with the permission to create projects (Administrator or Project Owner).

2. Right-click on the server name and select **Open Server Package File**. Open the CDS folder and select the `EDQ-CDS - Initialize Reference Data.dxi` file.
3. Expand the `EDQ-CDS - Initialize Reference Data.dxi` file and drag the whole **EDQ-CDS - Initialize Reference Data** project onto the **Projects** node.
4. Repeat steps 2 and 3 for the `EDQ-CDS.dxi` and the EDQ-CDS project it contains.
5. Repeat steps 2 and 3 for the `EDQ-CDS - Published Processors.dxi`.
6. Repeat steps 2 and 3 for the `EDQ-CDS - Data Quality Health Check.dxi` and the EDQ-CDS - Data Quality Health Check project it contains.
7. Once the projects have been imported, right-click on the `.dxi` files, and select **Close Package File**.

1.3 Configuring with Run Profiles

There are several configuration options for EDQ-CDS that are controlled by the properties in the EDQ-CDS Run Profiles that are installed with the product and used as follows:

File Name	Use	Property Sets
`edq-cds.properties`	Default EDQ-CDS Run Profile.	Language High Frequency Name Maps Key Generation (Real-time and Batch) Match Threshold Match Settings for disabling reference data input in real-time. Required for HA. Real-time Match Results Address Cleaning Staging Data for Batch Jobs Staged Data Visibility Match Settings Key Analysis Settings
`edq-cds-siebel.properties`	Sets properties specific to the Siebel EDQ-CDS integration.	Language High Frequency Name Maps Key Generation (Real-time and Siebel Batch) Key Analysis Settings Match Settings Real-time Match Results Address Cleaning Siebel Staging Data for Batch Jobs Staged Data Visibility

File Name	Use	Property Sets
edq-cds-data-quality-health-check.properties	Sets properties for Health Check functions.	EDQ Dashboard Source Input File Encoding Export Check Results Address Verification Country Code Individual Results Book Functionality Entity Results Book Functionality Staged Data Visibility
edq-cds-daas.properties	Not used at this time.	
edq-cds-fusion.properties	Not used at this time.	

These files are in the oedq.home/runprofiles directory of your EDQ installation directory. You can copy properties from one file to another so that the Run Profile you want to use contains all of the properties necessary to your configuration.

To edit a Run Profile:

1. Copy the Run Profile to the oedq.local.home/runprofiles directory of the EDQ installation and rename it.
2. In the oedq.local.home/runprofiles directory, open the Run Profile with a text editor.
3. Edit the values of the properties as required.
4. Save the file.

When an update occurs, files in the oedq.home/runprofiles directory will be over-written. The files in the oedq.local.home/runprofiles directory will not be affected.

The properties in each Run Profile fall into several categories, as described in the following sections.

Note : It is also possible to configure Address Cleaning on a per country basis, although this is not done using the Run Profile, see Section 1.3.4, "Address Cleaning Properties."

1.3.1 Pre-Initialized Reference Data

The initialized Latin reference data and the cdslists-initialized-full.zip file (pre-installed in the Oracle_Home/edq/oracle.edq/edq-cds/landingarea/cdslists/cdslists-initialized-full.zip directory) together contain initialized reference data for all supported languages.

To use initialized reference data for all other supported languages, extract the cdslists-initialized-full.zip file over the cdslists directory, overwriting pre-existing data.

To use a different set of languages (for example, only Japanese) or to customize the reference data (for example, to add additional name standardizations), prepare and initialize it as required. This overwrites the pre-prepared files.

Note : If this pre-initialized Reference Data is used, it is *not* necessary to use Section 1.3.2, "Initialize Reference Data Properties."

1.3.2 Initialize Reference Data Properties

The section explains how to configure the properties of the Initialize Reference Data project using run profiles.

1.3.2.1 Language Domains

By default, name data for all non-Latin script languages is excluded when using the Run Profile. This is controlled by the following property:

```
phase.Initialize.process.*.Language\ Domains = LAT
```

Note:

- This value is set to LAT by default, which means all Latin data is included. To exclude Latin data, delete this value.
- Multiple language domains can be specified as a comma-separated list.

To include data in one or more script languages, add the associated property value, as documented in the comments of the Run Profile.

For example, to include Arabic script data, add the ARA value to the property:

```
phase.Initialize.process.*.Language\ Domains = LAT, ARA
```

If you edit this property, you must run the Initialize Reference Data job.

1.3.2.2 High Frequency Names Only

By default, all names are included when records are processed. It is possible to exclude those non-Latin names that do not occur with a high frequency (for example, are not commonly used).

This is controlled by the following property:

```
phase.Initialize.process.*.High\ Frequency\ Only = N
```

To exclude uncommon non-Latin names, change this property value to `Y`.

If you edit this property, you must run the Initialize Reference Data job.

1.3.3 Matching Properties

These values are used to control key profile and matching behavior.

1.3.3.1 Key Profile

The `keyprofile` parameter attribute is used to specify the key methods to use for Key Generation in both batch and real-time. It can be set as follows, either to a pre-defined profile (*Loose*, *Strict* or *Typical*)

```
phase.Individual\ Keygen.process.*.keyprofile = Typical
phase.Batch\ Individual\ Match.process.*.keyprofile = Typical
phase.Entity\ Keygen.process.*.keyprofile = Typical
phase.Batch\ Entity\ Match.process.*.keyprofile = Typical
```

```
phase.Address\ Keygen.process.*.keyprofile = Typical
phase.Batch\ Address\ Match.process.*.keyprofile = Typical
```

or a manually encoded profile such as

```
AD112FNL5GNL5^10|GNW1FNL0^11|AD17AD25CTL10^12|FNM4PNL8^13|PNR6^14
```

If Legacy cluster methods are required (the old "clustering" methods prior to version 12.2.1) the following settings should be set in the run profile:

```
phase.*.process.*.uselegacykeygen = Y
```

and the levels set using

```
phase.Individual\ Keygen.process.*.clusterlevel = 2
```

1.3.3.2 Cluster Comparison Limits

The match processors contain default cluster comparison limits that are applied. When set, the cluster comparison limit is a default upper limit on the maximum number of comparisons to be performed on a single cluster. You calculate this figure by assessing the number of comparisons that you want performed in a cluster before processing it. If the number of comparisons that would be performed on the cluster is greater than the limit, the cluster is skipped.

You can set the limits for a given cluster by adding the cluster limits properties to your `edq-cds.properties` file and editing the limit values. For example:

```
# Change the cluster limits to have a maximum of 15,000 comparisons per cluster
group, and use the comparison limit in preference over the group limit.
phase.*.process.Match\ -\ Individual.*.individual_match_cluster_comparison_limit =
15000
phase.*.process.Match\ -\ Individual.*.match_cluster_group_limit = 0
phase.*.process.Match\ -\ Entity.*.match_cluster_comparison_limit = 15000
phase.*.process.Match\ -\ Entity.*.match_cluster_group_limit = 0
```

1.3.3.3 Batch Matching

This section describes properties for controlling batch matching:

- Batch Match Threshold
- Batch Compound Comparison and Weighting

1.3.3.3.1 Batch Match Threshold By default, the match threshold in the project for Batch processing of all record types is set to `70` (on a percentage scale). Matches with a rule score below this value will not be returned.

The match threshold to be used for batch matching is specified using the following run profile parameters. To set a different level for one or more types of processing, edit the values of the following properties accordingly:

```
######### Match Threshold ############
# Rule score below which matches will not be returned
# Default = 70 if this property is absent

# Batch Matching
phase.Individual\ Match.process.*.matchthreshold = 70
phase.Entity\ Match.process.*.matchthreshold     = 70
phase.Address\ Match.process.*.matchthreshold    = 70
```

Note : While the match thresholds set in the Run Profile override the default project settings, values passed from the Web Service take priority over both.

1.3.3.3.2 Batch Compound Comparison and Weighting Whether each compound comparison should be enabled for matching, and if so, what weighting it should use, can be controlled using the following run profile parameters:

```
phase.Individual\ Match.process.*.overallscore.XXX.weighting = 7
phase.Individual\ Match.process.*.overallscore.XXX.enabled   = Y
...
...
phase.Entity\ Match.process.*.overallscore.XXX.weighting = 10
phase.Entity\ Match.process.*.overallscore.XXX.enabled   = N
...
...
```

where `XXX` is the name of the compound comparison. This also applies to the new custom attributes.

1.3.3.4 Real-time Options

When using the real-time Key Generation or Matching services, it is possible to override certain options on a per-message basis, by passing in the settings in the header of the message. For example, in the real-time matching service, it is possible to override the `matchthreshold` (see Batch Match Threshold) on a per-message basis, you must pass it in on the message header rather than on every record. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dn="http://www.datanomic.com/ws">
  <env:Header/>
  <env:Body>
    <dn:request matchthreshold="80">
      <dn:record>
...
```

For real-time key generation it also possible to set the keyprofile option to use on a per-message basis, for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dn="http://www.datanomic.com/ws">
  <env:Header/>
  <env:Body>
<dn:request
  keyprofile="Strict"
>
```

Similarly, the other match-related run profile settings relating to compound comparison weighting and enablement can also be specified on a per-message basis, for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dn="http://www.datanomic.com/ws">
  <env:Header/>
  <env:Body>
<dn:request
```

```
   overallscore.address.enabled="Y"
   overallscore.address.weighting="21"
>
      <dn:record>
...
```

> **Note:** When using the Web Service Tester tool there is a header input field provided, where any of these settings can be input in a simple key value pair input (for example, matchthreshold="80").
>
> For a full list of properties that can be specified in the header of a real-time request see the following tables: Table 5–1, Table 5–2, and Table 5–3.

1.3.3.5 Match Results Configuration for Real-time Jobs

Siebel 8.1 and later requires that real-time matching responses include both the driving record and all matching candidate records, with their match scores. For all other use cases it is not necessary to return the driving record in the response. The following option controls whether or not to include the driving record in responses to real-time matching services:

`phase.*.process.*.Return\ Real-time\ Driving\ Record=`

The default settings for this property are as follows:

- `edq-cds.properties - N`
- `edq-cds-siebel.properties - Y`

If this option is set to `Y` the driving record (with only the ID populated) is returned as the first record in the response, where there was at least one match in the candidate set. Otherwise, the driving record is excluded.

1.3.4 Address Cleaning Properties

When using the Address Cleaning service with EDQ-AV, the properties described in this section can be configured as required. For more information about Address Cleaning, see *Oracle Enterprise Data Quality Address Verification Installation Guide*.

1.3.4.1 Default Country Code

`phase.*.process.Clean\ -\ Address.Default\ Country\ Code = US`

This property can be used to define a system-level default country code in installations where addresses will typically all be in the same country and will not be specified per request on the interface.

The default value is `US`. Any codes that are entered here are expected to comply with the ISO-3166-1-alpha-2 specification.

1.3.4.2 Whether Address Verification Should Enable Geocoding

`phase.*.process.Clean\ -\ Address.Enable\ Geocoding = Y`

This property controls whether the Address Verification processor should use Geocoding, and correspondingly return latitude and longitude information with the cleaned address.

1.3.4.3 Default Allowed Address Verification Result Codes

```
phase.*.process.Clean\ -\ Address.Default\ Allowed\ Verification\ Result\
Codes = PV
```

This property specifies which Verification codes are permitted, which by default are `P`(partially verified) and `V`(verified).

1.3.4.4 Default Minimum Address Verification Level

```
phase.*.process.Clean\ -\ Address.Default\ Minimum\ Verification\ Level =
2
```

This property specifies the minimum required (post-process) Verification Match level, on a scale of `1` to `5`. The default value is `2`.

1.3.4.5 Default Minimum Address Verification Match Score

```
phase.*.process.Clean\ -\ Address.Default\ Minimum\ Verification\ Match\
Score = 95
```

This property specifies the minimum Match score required, on a scale of `1-100`. The default setting is `95`.

> **Note :** The three properties above set system-level defaults that control whether the Address Verification processor should actually clean an address based on the strength of the verification it is able to perform. These properties can also be overridden on a per-request basis by specifying them on the Address Cleaning interface, or overridden on a per-country basis (see Section 1.3.7, "Address Cleaning Per Country.")

1.3.4.6 Number of Lines Returned by the Address Clean Process

```
phase.*.process.Clean\ -\ Address.Number\ Of\ Address\ Lines =
```

Applications commonly support two, three or four address lines for the house number/street part of the address.

This property indicates the number of cleaned address lines that should be returned by the cleaning service.

The default settings in the Run Profiles are as follows:

- `edq-cds.properties` - `4`
- `edq-cds-siebel.properties` - `2`

1.3.4.7 Post-Processing

Post-processing is run after address cleaning, to apply certain changes to the results which have been returned from AV. This functionality is intended for Siebel integrations. Therefore, the default settings in the Run Profiles are:

- `edq-cds.properties` - `N`
- `edq-cds-siebel.properties` - `Y`

Standardize a Verified Country Name to Specific Values

If this value is set to `Y` `country` names are standardized to those in the default Siebel pick list:

```
phase.*.process.Clean\ -\ Address\ Post\ Process.Standardize\ Verified\
Country\ to\ CRM\ Values =
```

Standardize a Verified `adminarea` to Specific Values

If this value is set to `Y`, only `adminarea` values in the default Siebel pick list are returned:

```
phase.*.process.Clean\ -\ Address\ Post\ Process.Standardize\ Verified\
Admin\ Area\ to\ CRM\ Values =
```

If this value is set to `N`, the `adminarea` value of the output address is passed back to Siebel.

> **Note :** The default Siebel pick list only includes US states. The reference data set that controls this in CDS is Address Clean - Admin Area to Standard CRM Admin Area. Entries added to this list are added to the Admin Areas that are considered 'valid' for pass-back to a CRM application such as Siebel that uses bound lists for the mapped field.

Standardize Blank Verified Address Fields to be Returned as a Space

When the Siebel Data Quality interface receives back an empty string from a standardization service, it interprets this as meaning 'the current value should be retained'. In the case of Address Cleaning, it is sometimes desirable deliberately to remove the current value for an attribute; for example, an address standardization service may change an input address such that sub-building details are moved from the second line of the address to the end of the first line. In this case, in order not to duplicate the sub-building details in both address lines, a single space is returned in a return attribute to indicate to Siebel that the input value should be removed. Siebel does not in fact insert a space into the value; it interprets the space as meaning the value should be removed.

If this value is set to `Y`, any blank fields are populated with a single space character before being returned to Siebel:

```
phase.*.process.Clean\ -\ Address\ Post\ Process.Standardize\ Verified\
Blank\ Address\ Fields\ to\ Space =
```

1.3.5 Staging Data Configuration Parameters for Batch Jobs

By default, the Staging Data configuration for Batch jobs is derived from the candidate snapshots and the properties are set using the defined data source and the table names are set to the EDQ-CDS defaults. These properties can be edited as necessary if you want to point the (generic) batch matching jobs at different staging tables. The `SERVERID` and `JOBID` columns are used to enable processing of multiple batch jobs in parallel so they need to be edited in the run profile accordingly prior to each job submission; if they are not needed then default values can be used.

```
######### Staging Data Configuration Parameters For Batch Jobs ############
# The JNDI data source name and table names may be different dependent on the
installation

# Where clause for candidate snapshots, to obtain data for specific server and job
phase.*.snapshot.*.where   = serverid = 'SERVERID' AND jobid = 'JOBID'

# Export parameters for specific server and job
phase.*.process.*.serverid = SERVERID
```

```
phase.*.process.*.jobid    = JOBID

# JNDI data source name for staging schema in database
phase.*.snapshot.*.remotejndi = jdbc/edqcdsstaging
phase.*.export.*.remotejndi   = jdbc/edqcdsstaging

# Table names for candidate staging tables (snapshots)
phase.*.snapshot.Entity\ Candidates.table_name     = EDQCDS_CANDIDATES_ENT
phase.*.snapshot.Individual\ Candidates.table_name = EDQCDS_CANDIDATES_IND
phase.*.snapshot.Address\ Candidates.table_name    = EDQCDS_CANDIDATES_ADD

# Table names for result staging tables (exports)
phase.*.export.Batch\ Matches.table_name           = EDQCDS_MATCHES
phase.*.export.Batch\ Key\ Generation\ Results.table_name = EDQCDS_CLUSTER_KEYS
```

1.3.6 Staged Data Visibility Settings within Server Console

By default, most Staged Data sets are suppressed in the Results view of the Server Console. Only those Staged Data sets listed in this section of the Run Profile are visible in Server Console by default:

```
# Initialize Project
stageddata.\[QA\]\ Single\ chars.visible = yes
stageddata.\[QA\]\ Variant\ has\ Multiple\ Masters.visible = yes
stageddata.\[QA\]\ Variant\ is\ Master.visible = yes
stageddata.Conflict\ Res\ \-\ Removed\ Links\ ALL.visible = yes
```

To make other Staged Data sets visible, add a property in the format of those included in the Run Profile, as in the preceding example.

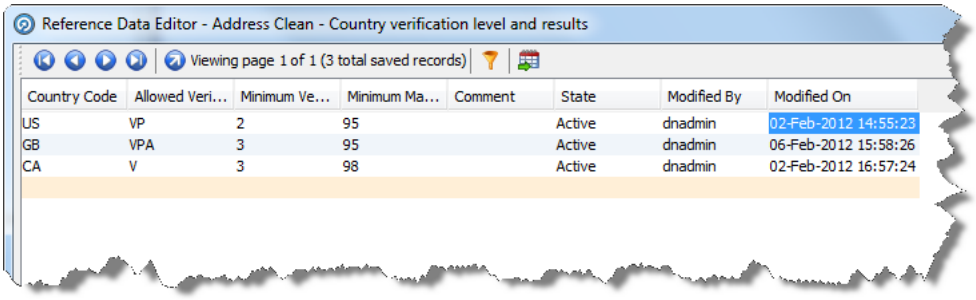
1.3.7 Address Cleaning Per Country

The extent to which EDQ-AV can verify addresses varies depending on the country. Additionally, address data from certain countries may be trusted more than data provided for others.

To allow for this, it is possible to set different parameters for address cleaning on a per-country basis.

To set the required parameters:

1. Open the Director client.
2. In the Project Browser, select **EDQ-CDS > Reference Data**.
3. Open the **Address Clean - Country verification level and results** Reference Data.



4. In the Reference Data Editor, change the default settings for US, GB and CA, and add additional rows and settings for other countries as required.
5. Click **OK** to save changes, or **Cancel** to abandon.

Note : For further details of the Verification settings, see Chapter 5, "Using Business Services."

1.4 Initializing Custom Reference Data

If the pre-initialized Reference Data shipped with EDQ-CDS is used, this procedure is not required. However, if any of the initialization options detailed in Section 1.3.2, "Initialize Reference Data Properties" have been changed from their default settings the Reference Data must be re-initialized by running the job in the Server Console.

To do this, use the following procedure:

1. Open the Server Console.
2. Expand the **EDQ-CDS - Initialize Reference Data** project.
3. Right-click the **MAIN Initialize Reference Data** job and select **Run...**
4. Select the EDQ-CDS run profile and specify a Run Label of **cds**.

Note:

- This job must be re-run if the Reference Data is customized, or if the Run Profile is modified in order to select different languages to initialize.
- Oracle recommends that `cds` is used as the Run Label for all CDS jobs.

1.5 Starting and Stopping Real-time Jobs and Processes

There are several jobs that *must* be running in order to use the Real-time processes. These jobs are controlled by two other jobs: **Real-time START ALL** and **Real-time STOP ALL**, which *must* be started in the Server Console.

Note : By default, all CDS Real-time services are started by the **Real-time START ALL** and **Real-time STOP ALL** jobs.

For optimum performance efficiency, do not run services that you are not using. For example, if you are not using the cleaning services you can remove the triggers that start them. In some cases, such as smaller servers, it may be required to remove the triggers that start the unused services from the **Initiate Real-time Services** phase of the **Real-time START ALL** job. To remove a trigger, right-click on the phase, select **Configure**, and remove the job that you do not need. Both Key Generation and Match Jobs must be running for a matching service.

To start the Real-time processes:

1. Open the Server Console.
2. Expand the **EDQ-CDS** project.
3. Run the **Real-time START ALL** job.
4. Select the required Run Profile from the drop-down field.

> **Note :** If running the job in order to provide services to Siebel (either CRM or UCM), the edq-cds-siebel Run Profile must be selected, so that the correct configuration settings for Siebel are used.
>
> If running the job to provide services to other applications, the edq-cds Run Profile is recommended. For more information, see Section 1.3, "Configuring with Run Profiles."

5. Enter **cds** as the Run Label.
6. Click **OK**.

Under certain circumstances it may be necessary to stop and restart the Real-time processes. For example, if new Reference Data has become available, it will be necessary to stop the Real-time processes, re-run the **Initialize Reference Data** job, and start the Real-time processes again.

To stop the Real-time processes:

1. Open the Server Console.
2. Expand the **EDQ-CDS** project.
3. Run the **Real-time STOP ALL** job.

1.5.1 Scheduling a Real-time START ALL Job at Start Up

If the server restarts, it will be necessary to also restart the Real-time jobs with the appropriate Run Profile and Run Label. To ensure this happens automatically, use the following procedure to configure the **Real-time START ALL** job to run at start up:

1. Open the Server Console
2. Expand the EDQ-CDS project.
3. Open the **Real-time START ALL** job
4. Right click and select the **Schedule** option.
5. Select the **Startup** radio option.
6. Select the required Run Profile from the drop-down field.

> **Note :** If running the job in order to provide services to Siebel (either CRM or UCM), the **edq-cds-siebel** Run Profile must be selected, so that the correct configuration settings for Siebel are used.
>
> If running the job to provide services to other applications, the `edq-cds` Run Profile is recommended.

7. Specify a **Run Label** of **cds**.
8. Click **OK** to save the changes.

2

Customizing Customer Data Services Pack

This chapter describes how EDQ-CDS can be customized to take advantage of some of the more advanced features of the product.

This chapter includes the following sections:

EDQ-CDS has been designed to perform well with minimal customization. Ready-to-use, the application can perform key generation and matching of individual, entity and address data in connected supported applications with little or no configuration changes required.

2.1 Using Stand-Alone Batch Matching

EDQ-CDS is designed to process customer data from any external system or stand-alone source. By default, pre-configured batch jobs are provided that work with a set of staging tables. Reconfiguring the product to process data from other sources, such as a text file, is straightforward.

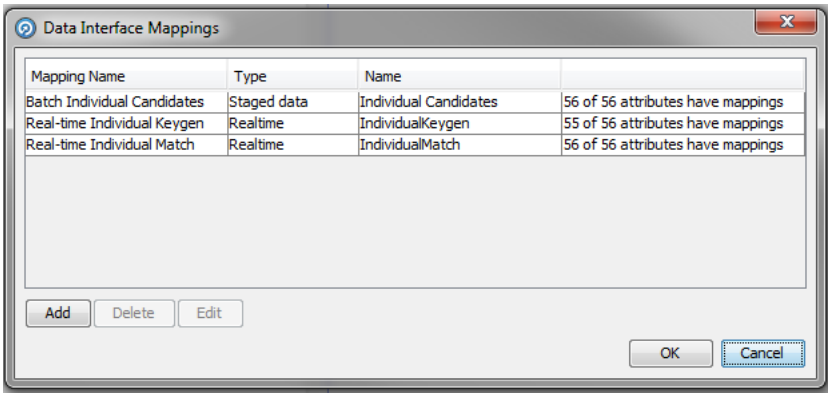
In order to reuse the batch data matching services provided, it is necessary to create new input and output mappings for the data interfaces. The following sections use examples that demonstrate how to do this and how to run matching using a modified copy of an existing job configuration.

2.1.1 Using Stand-Alone Individual Batch Matching

You can create a new stand-alone individual batch matching job using the following example steps:

1. Ensure that no jobs are currently running.
2. In the EDQ-CDS project, create a new server-side data store named **File In: Individuals** that points to the structured text file containing the customer data to be processed. It is important that this is created as a server-side data store in order to be used within a job definition.
3. Create a new snapshot named **Individuals** using the **File In: Individuals** data store as a source.
4. Create the Input Data Interface mappings as follows:

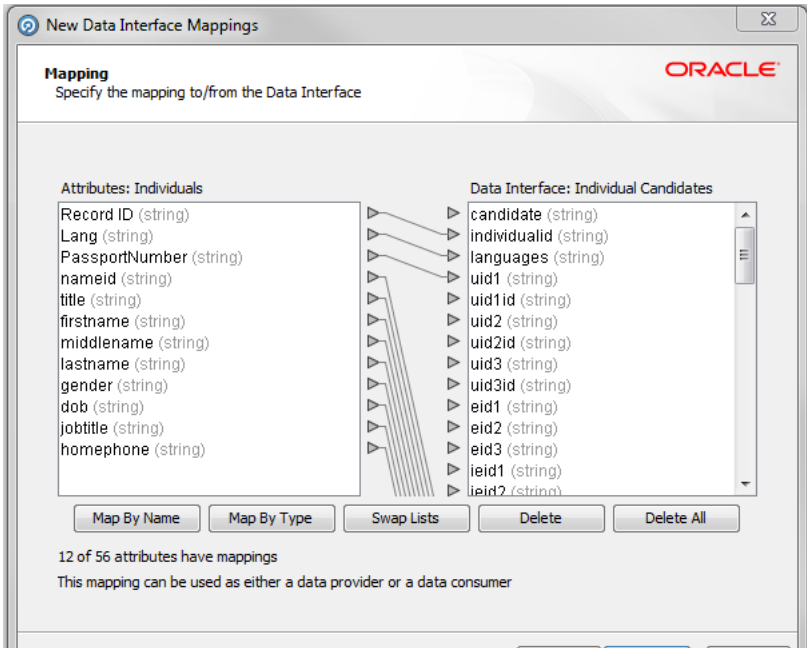
a. Right-click the **Individual Candidates** data interface and select **Mappings...** to open the **Data Interface Mappings** dialog.



b. Click **Add** to open the **New Data Interface Mappings** dialog.

c. Select the **Individuals** snapshot as the source and click **Next**. The Staged data default type is used.

d. Map the Customer Data Attributes on the left of the dialog to the Data Interface Attributes on the right as follows:



Note : In some instances, it may be necessary to construct a process that reads from the snapshot and reshapes the data to match the Data Interface, see Section 2.1.2, "Converting Data to the Interface Format."

---

e. Click **Next**.

f. Name the data interface mapping **Individual Candidates** and click **Finish** to save.

g. Click **OK**.

5. Create a new Staged Data named **Individual Matches** with columns corresponding to the columns in the Matches data interface.

6. Create the Output Data Interface mappings as follows:

   a. Right-click the **Matches** data interface and select **Mappings...** to open the **Data Interface Mappings** dialog.

   b. Click **Add** to open the **New Data Interface Mappings** dialog.

   c. Select the **Individual Matches** staged data as the target and click **Next**.

   d. Map the **Matches** data interface attributes on the left to the **Individual Matches** attributes on the right as required.

   e. Click **Next**.

   f. Name the Data Interface mapping **Individual Matches** and give it a description, then click **Finish**.

   g. Click **OK** to close the dialog.

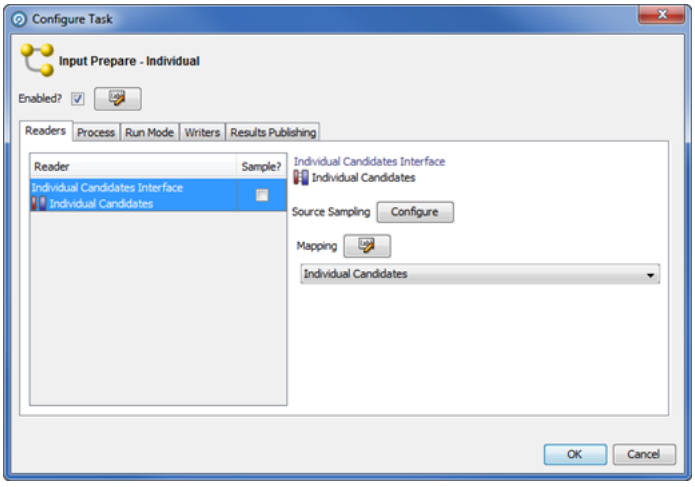
7. Create a new server-side delimited text data store called **File Out: Individual Matches** to use as a target for the match results. Alternatively, the data can be written to a database if required.

8. Create a new export called **Matches to File Out: Individual Matches** that uses the **Matches** data interface as the source to export from, and the **File Out: Individual Matches** as the target for the export.

9. Create and configure a job to run matching as follows:

   a. Create a copy of the **Batch Individual Match** job, rename it **Batch Individual Match using Text File**, and then open it.

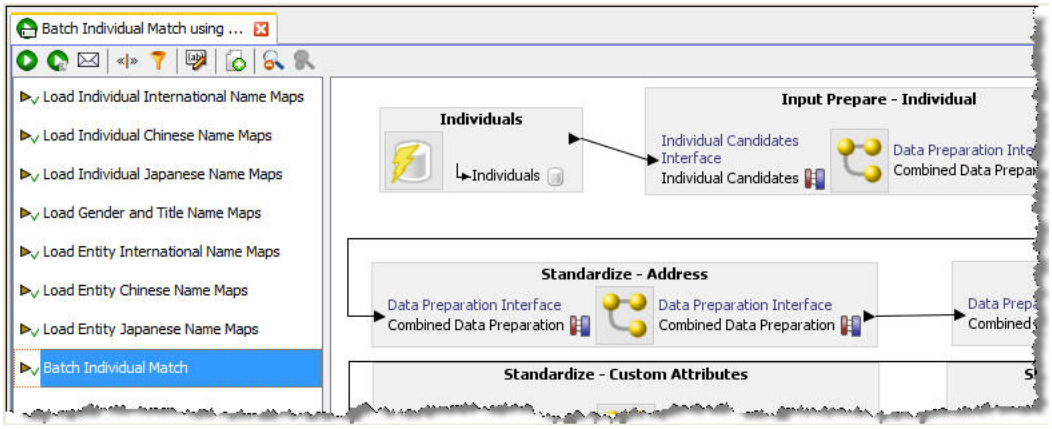
   b. Open the **Individual Match** job phase, change the source of the input data by double-clicking on the **Individual Candidates** data interface and selecting the **Individual Candidates** mapping.



   c. Click **OK** to apply the changes. The job configuration is modified accordingly and the old snapshot and staged data items are disconnected.

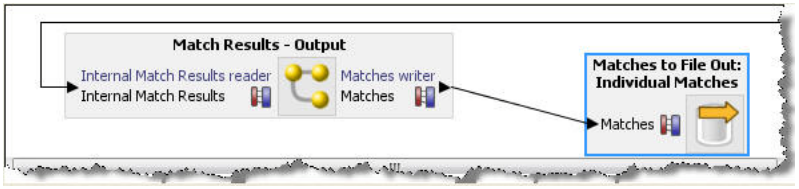
d. Delete the **Individual Candidates** snapshot task.

e. Drag the **Individuals** snapshot from the **Snapshot** in the **Tool Palette** into the open job phase and make sure it is connected to the **Individual Candidates** mapping.



f. Drag the **Matches to File Out: Individual Matches** export task from the **Export** in the **Tool Palette** into the open job phase and connect it to **Match Results - Output**.

g. Delete the **Batch Matches** export task.



10. Close the job and save the configuration changes.

2.1.2 Converting Data to the Interface Format

It may not always be possible to directly map the input source to the candidates interface if:

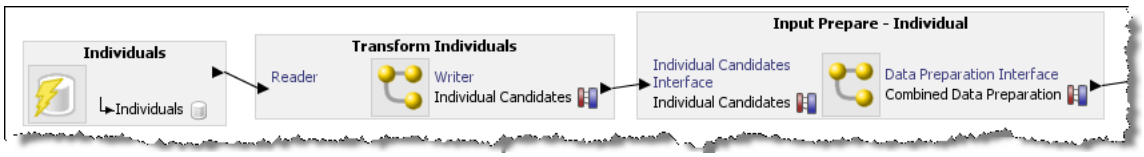
- fields are of the wrong data type (for example, "Date of Birth" in a date field); or
- fields need transforming to a compatible format/structure (for example, Individual names in a full name field).

If this is the case, then the input data should be run through a custom EDQ process to convert the data as appropriate as in the following example steps:

1. Ensure that no jobs are currently running.
2. Create a data store and snapshot for the input data as in steps 2 and 3 from Section 2.1, "Using Stand-Alone Batch Matching."
3. In the EDQ-CDS project, right-click the **Processes** node in the Project Browser and select **New Process...** to open the **New Process** wizard.
4. Select the snapshot created in step 2 as the data source.
5. Click **Next**.
6. On the last page of the wizard, rename the process **Transform Individuals**, then click **Finish** button to create the process.

7. On the Process canvas, add the necessary processors to transform the data to the interface format. For example, use a **Convert Date to String** processor to convert a date of birth in date format to the required format for the Candidates interface (for example, either yyyyMMdd, MM/dd/yyyy, yyyy-MM-dd or dd-MMM-yy).
8. Add a Writer processor to the process canvas and connect it to the process data stream:

9. In the **Writer Configuration** dialog, select the **Individual Candidates** data interface and map the attributes accordingly.
10. Create and configure a new job as follows:
    a. Make a copy of **Batch Individual Match** job, renaming it **Batch Transformed Individual Match**.
    b. Open the new job.
    c. Double-click on the **Individual Match** job phase.
    d. Use steps 9.d. - 10 of Section 2.1, "Using Stand-Alone Batch Matching" from step 9.d onwards, adding in the new Transform Individuals process between the Individuals snapshot and the process Input - Prepare - Individual. The resulting job should look like the following:



2.2 Using Cleaning Services

The cleaning processes provided with EDQ-CDS are provided as templates only, with the exception of the Address Cleaning process which is fully functional and uses EDQ-AV for address verification and standardization. The Individual and Entity cleaning processes are intended to be customized to meet the data standardization requirements of the implementation.

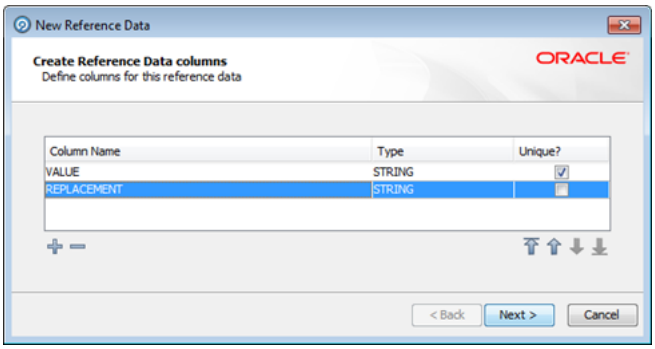
2.2.1 Customizing the Cleaning Services

The examples in the following sections demonstrate modifying the cleaning services provided with EDQ-CDS.

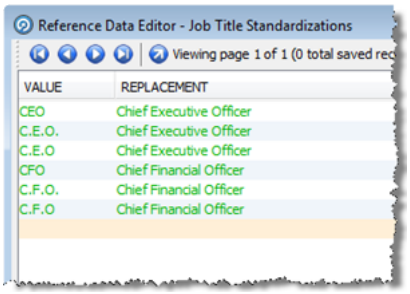
2.2.1.1 Standardizing Job Titles

Modify the Individual Cleaning service to standardize job titles as in the following example steps.

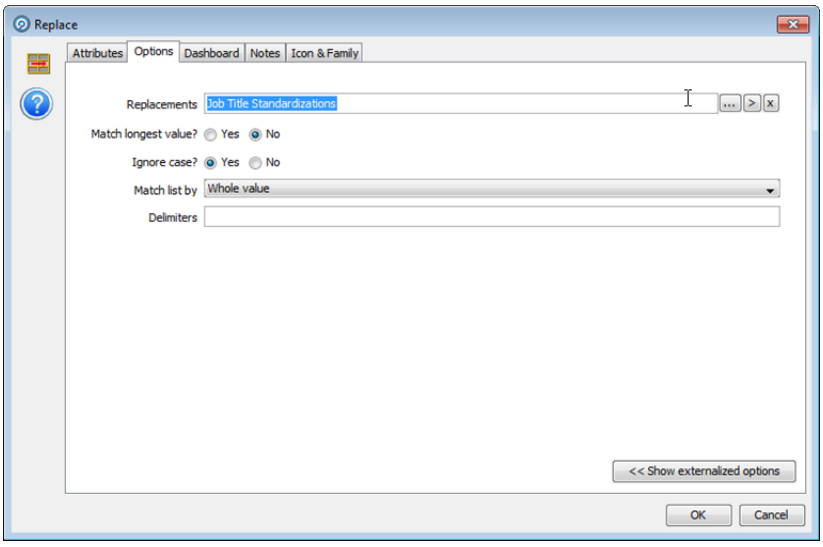
1. Ensure that no jobs are currently running.
2. In the EDQ-CDS project, create a new Reference Data set with the columns as follows:



3. Click **Next** through the **New Reference Data** wizard with the name **Job Title Standardizations**.
4. Click **Finish** to close the wizard. The **Reference Data Editor** dialog opens.
5. Add the required job title standardizations; for example:

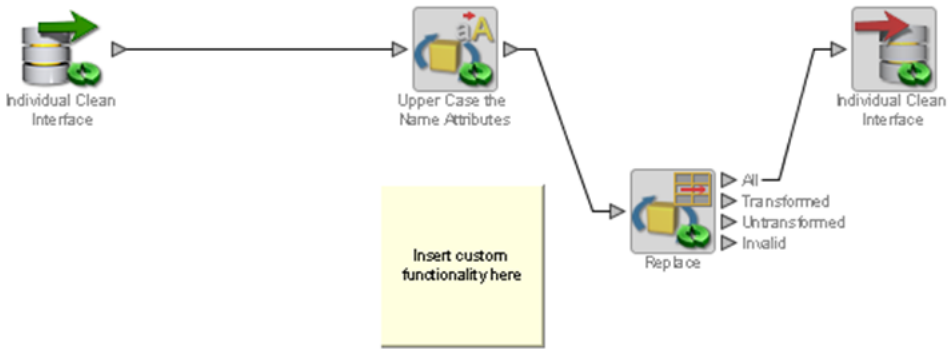


6. Open the **Clean - Individual** process.
7. Add a new **Replace** processor to the Process Canvas and connect it to the output of the **Upper Case the Name Attributes** processor.
8. In the **Processor Configuration** dialog, set the **jobtitle** attribute as the Input field, and on the Options tab select the **Job Title Standardizations** Reference Data in the **Replacements** field.



9. Click **OK** to close the processor configuration dialog.

10. Connect the **All** output of the **Replace** processor to the **Writer**, then click **OK** without making any changes to the **Writer** configuration.
11. On the Process Canvas delete the direct link between the **Upper Case** processor and the **Writer**.



12. Close the process and save the changes.
13. Test the modified cleaning service.

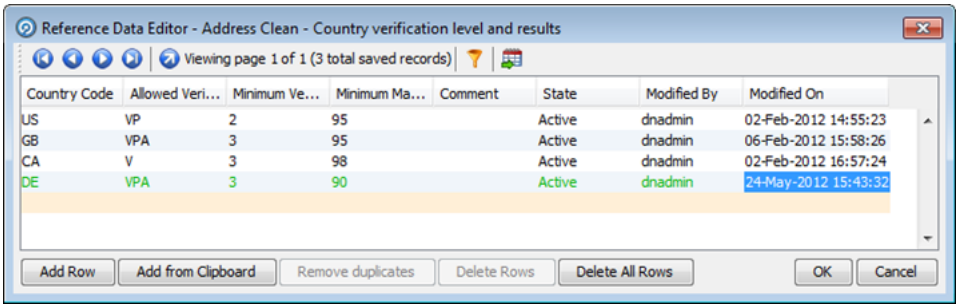
2.2.2 Changing Country-Specific Address Cleaning Settings

The default settings (Allowed Verification Results, Minimum Verification Level and Minimum Match Score) used in the Address Cleaning process that uses EDQ-AV can be overridden on a per-country basis by simply modifying reference data.

2.2.2.1 Reducing the Strictness of German Address Validation

Modify the EDQ-AV settings to reduce how strictly German addresses will be validated as in the following example steps.

1. Ensure that no jobs are currently running.
2. In the **EDQ-CDS** project edit the **Address Clean - Country verification level and results** Reference Data.
3. Add the following row:
   - Country Code: DE
   - Allowed Verification Results: VPA
   - Minimum Verification Level: 3
   - Minimum Match Score: 90



4. Click **OK** to close the dialog.

2.3 Adjusting Matching

This section explains how you can change the EDQ matching settings.

2.3.1 Changing the Key Method To Use During Matching

Keys are used as the first stage of matching to pre-select similar records. This will happen inside EDQ for batch matching, or in the calling application during candidate selection for real-time matching.

By default, the key methods that are used during matching depends on the value of the `keyprofile` setting. The key profile specifies the enablement of key methods, allowing EDQ-CDS to offer a wider menu of key method algorithms.

The methods for controlling which Match key methods are used differs for Batch and Real-Time processing. The following sections contain examples to show you how to modify the key methods used.

2.3.2 Reviewing Matches in EDQ

The EDQ-CDS Matching services return only those records that matched with a score equal to or greater than the `matchthreshold` setting, and for those records it only returns the record ID, rule name and score. It is useful to be able to view the full record details during rule tuning in order to analyze matches. The Match Review application is a helpful tool in this process.

2.3.2.1 Enabling Match Review in Individual Batch Matching

You can enable match review for individual batch matching as in the following example steps.

1. Ensure that no jobs are currently running.
2. In the **EDQ-CDS** project, open the **Match - Individual** process.
3. Double-click on the **Match Individuals** processor to open the **Match Configuration** dialog.
4. Click **Advanced Options**.
5. From the Review System list, select **Match Review**, and then click **OK**. This makes the Assign Relationship Review option active.
6. Click **Assign Relationship Review**.
7. In the dialog displayed, select the appropriate user or user group in the **Assigned To** drop-down field.
8. Click **OK** to close the dialog.
9. Close the process and save the configuration changes.
10. Open the **Batch Individual Match** job.
11. Locate the Match phase, right-click on the **Match Prepare** task and select **Configure**. The **Task Configuration** dialog opens.
12. Select the **Process** tab, and check the **Enable Sort/Filter in Match**? option.
13. Click **OK** and close the job, saving changes when prompted.
14. Run the job from Director with the appropriate run profile and no run label to regenerate the data.

Note: In order to generate Match Review data, you must run jobs without a run label.

Matches can be reviewed as follows:

1. On the Launchpad page, click **Match Review** icon.

Note : If this application is not visible then you will need to publish it via the launchpad server configuration pages.

2. Login as a user with the appropriate security permissions (for example, a user that is a member of the group selected in step 5).
3. Select **Match - Individual** in the **Reviews** list in the left-hand panel to view the Match Review statistics.
4. Click the **Launch Review Application** link to start reviewing matches for the selected Review.

2.4 Modifying Reference Data Used in Matching

This section explains how you can modify your data to improve matching and provides examples to aid you.

2.4.1 Stripping Words/Phrases from Name Fields

It is possible to customize the system to strip certain words and phrases from names that are deemed to be noise and/or add little information, and therefore may lead to potential missed matches.

2.4.1.1 Removing Noise from Individual Names

Name fields in customer data systems are often overfilled with additional (non-name) information, either because there are no other suitable fields available or due to errors made by Data Entry users. Common examples include "Fred SMITH (DO NOT CALL)" and "John DOE (DECEASED)". This extraneous information can be removed during name standardization when a "distilled" name is created for use in matching.

Use the following example steps to remove noise from individual names:

1. Ensure that no jobs are currently running.
2. In the **EDQ-CDS - Initialize Reference Data** project open the **Strip List – Titles Latin** Reference Data.
3. Add the following rows to the Reference Data set:
   - DO NOT CALL
   - DECEASED
4. Click **OK** to close the dialog.
5. Re-run the **MAIN Initialize Reference Data** job from the Server Console to re-prepare the Reference Data files that are used by the Matching services.

Note : The Real-Time services will use the modified Reference Data sets the next time the full **Real-time START ALL** job (which re-snapshots the prepared Reference Data from files) is run.

To remove words and phrases from individual names in non-Latin scripts use the reference data **Strip List – Individual Script Strip List** Reference Data . This Reference Data set is used as a replacement map and should have a blank value in the second column.

2.4.1.2 Removing Noise from Entity Names

Noise words and phrases or common business words (including suffixes) in Entity names that add little value in matching can be removed during name standardization when a "distilled" name is created. An example of such a noise word is "International", which is often found in organization name fields.

Due to the high frequency of occurrence of this term it is often omitted or shortened when entering the name, which may lead to potential matches being missed. Therefore it may be more appropriate to remove the term and all known variants for the purposes of matching.

Use the following example steps to remove noise from entity names:

1. Ensure that no jobs are currently running.
2. In the **EDQ-CDS - Initialize Reference Data** project open the **Strip List – Entity Latin** Reference Data.
3. Add the following rows to the Reference Data set:
   - INTERNTL
   - INTL
   - INT
4. Click **OK** to close the dialog.
5. Re-run the **MAIN Initialize Reference Data** job from the Server Console to prepare the data.

To remove words and phrases from entity names in non-Latin scripts use the **Strip List – Entity Script Suffixes** Reference Data.

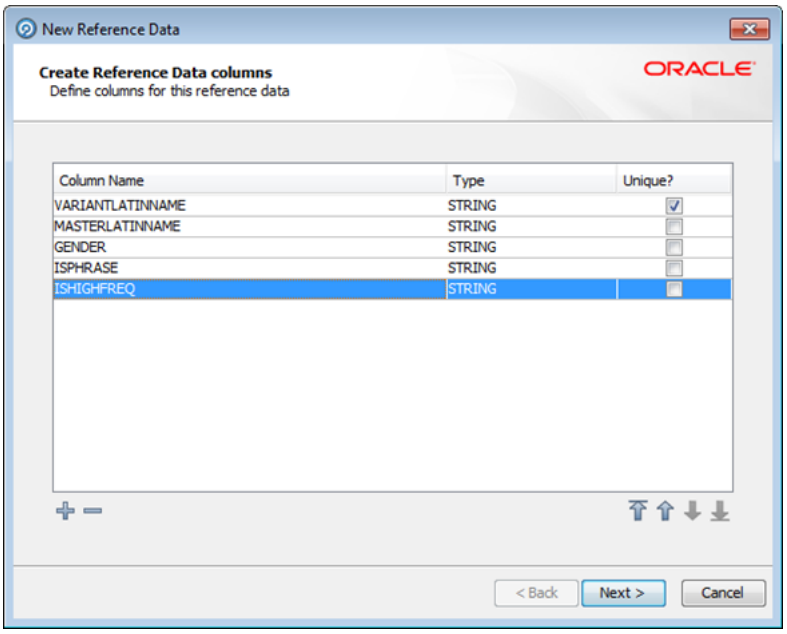
2.4.2 Changing Name Standardization

EDQ-CDS uses a name standardization technique in order to match name variants. It is supplied with a large collection of common name variants for various language domains. It is possible to customize these lists.

Note : If a name standardization is changed or added, the subsequent results may be eliminated during Conflict Resolution. For further details, see Section 2.4.3, "Resolving Conflicts".

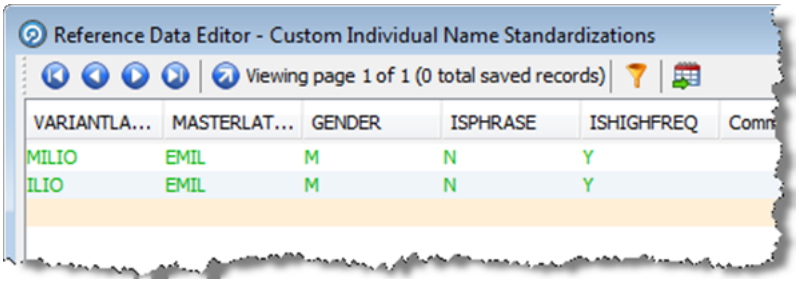
2.4.2.1 Adding Individual Name Standardizations

1. Ensure that no jobs are currently running.
2. In the **EDQ-CDS - Initialize Reference Data** project create a new Reference Data set with columns as in the following:



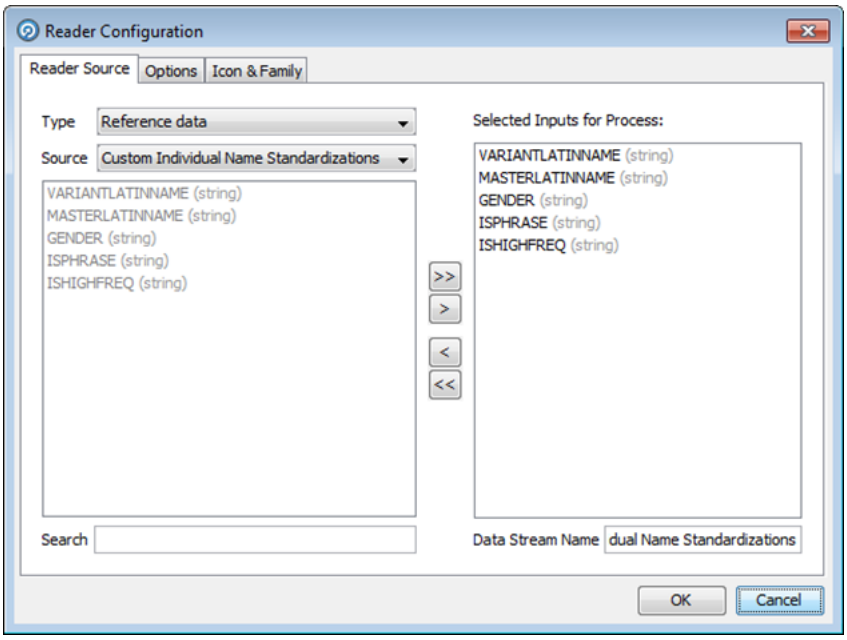
3. Click **Next** through the **New Reference Data** wizard and name it **Custom Individual Name Standardizations**.
4. Click **Finish** to close the dialog.
5. The **Reference Data Editor** dialog will open. Add the required name standardizations, where:
   - VARIANTLATINNAME is the name to be standardized.
   - MASTERLATINNAME is the standardized version of variant name.
   - GENDER takes the value M for male, F for Female, or U for unknown or ambiguous.
   - ISPHRASE takes the value N for single token names and Y for multi-token names containing whitespace.
   - ISHIGHFREQ is set to Y.

   Note : It is important to ensure that data is entered in upper case and that variant names only have a single master across all language domains.



6. Click **OK** to close the dialog.
7. Open the **[D] Initialise Individual Latin to Latin Data** process.

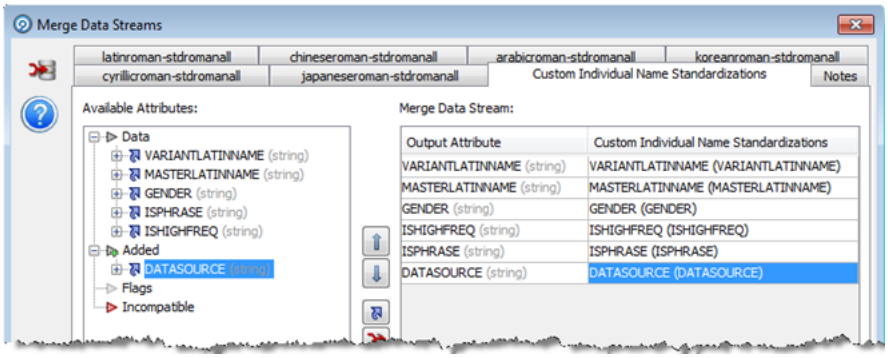
8. Add a **Reader** process to the Process Canvas and configure it to use the **Custom Individual Name Standardizations** Reference Data as the source, selecting all attributes for input to the process.



9. Add a new **Add String Attribute** processor to the process canvas and connect the reader to the new processor. In the processor configuration dialog rename the new attribute DATASOURCE and set the attribute value to CUSTOM.

10. Connect the output of the **Add String Attribute** processor to the **Merge Data Streams** processor.

11. In the **Custom Individual Name Standardizations** tab of the **Processor Configuration** dialog associate the Available Attributes with the Output Attributes in the **Merged Data Stream** area:



12. Click **OK** to close the dialog.

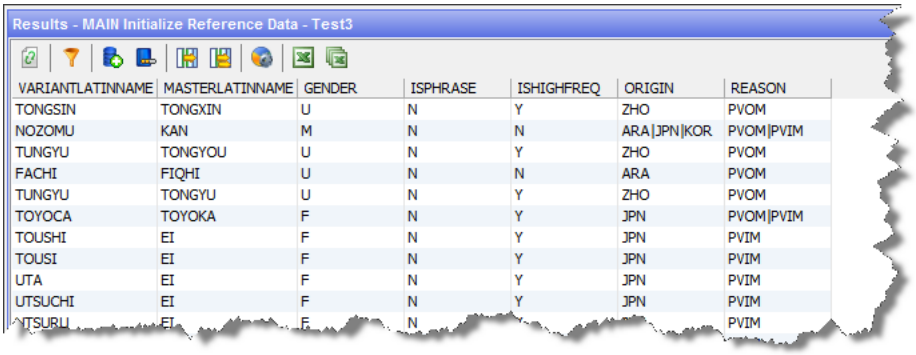
13. Close the process and save the configuration changes.

14. Re-run the **MAIN Initialize Reference Data** job from the Server Console to prepare the data.

2.4.3 Resolving Conflicts

Conflict resolution is performed to resolve issues arising when name standardization rules try to standardize names to more than one Master name. For example, if there is a rule that maps "Jon" to a Master of "John" and another that maps "Jon" to "John-Boy", there is a conflict. This conflict is resolved by assessing the importance of each Master name in the given standardization data. The best candidate is then selected as the primary Master, and other standardization maps conflicting with it are removed and quarantined.

As part of conflict resolution, each removed record is assigned one or more Reason Codes explaining why it is in conflict. These codes are displayed in the **REASON** column in the Server Console **Results** window:



VARIANTLATINNAME	MASTERLATINNAME	GENDER	ISPHRASE	ISHIGHFREQ	ORIGIN	REASON
TONGSIN	TONGXIN	U	N	Y	ZHO	PVOM
NOZOMU	KAN	M	N	N	ARA|JPN|KOR	PVOM|PVIM
TUNGYU	TONGYOU	U	N	Y	ZHO	PVOM
FACHI	FIQHI	U	N	N	ARA	PVOM
TUNGYU	TONGYU	U	N	Y	ZHO	PVOM
TOYOCA	TOYOKA	F	N	Y	JPN	PVOM|PVIM
TOUSHI	EI	F	N	Y	JPN	PVIM
TOUSI	EI	F	N	Y	JPN	PVIM
UTA	EI	F	N	Y	JPN	PVIM
UTSUCHI	EI	F	N	Y	JPN	PVIM
UTSURU	EI	F	N			PVIM

The Reason Codes are as follows:

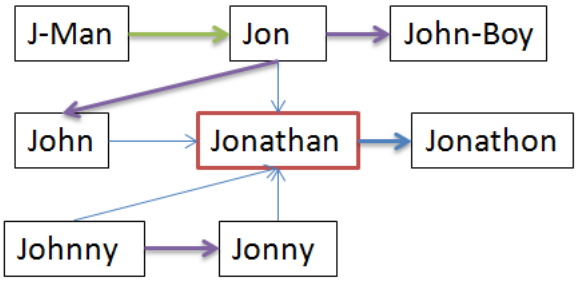
- **PIV**: The Primary record of a cluster of records (for example, the best Master identified for a set of equivalences) is also present as a variant to other Masters. All the instances where this Primary name is a variant are removed.
- **PVOM**: The records that are variants of the current Primary are also variants of other Masters. All the records for these variants pointing to other Masters are removed.
- **PVIM**: The records that are variants of the current Primary are also Masters to other variants. All the records where this variant is a Master are removed.
- **PIVCUTOFF**: Whereas the other removals take place after identification of Primary clusters, there comes a time where it is not efficient to continue to identify the Primaries, and the remaining records where the Master name also exists as a variant have all the variant versions removed in a final cull of records that violate integrity.

Expanding on the simple example given at the beginning of this section, let us assume that there are the following name standardization rules:

Master	Primary
J-MAN	JON
JOHN	JONATHAN
JOHNNY	JONNY
JON	JOHN
JON	JONATHAN
JON	JOHN-BOY
JONNY	JONATHAN

Master	Primary
JONATHAN	JONATHON
JOHNNY	JONATHAN

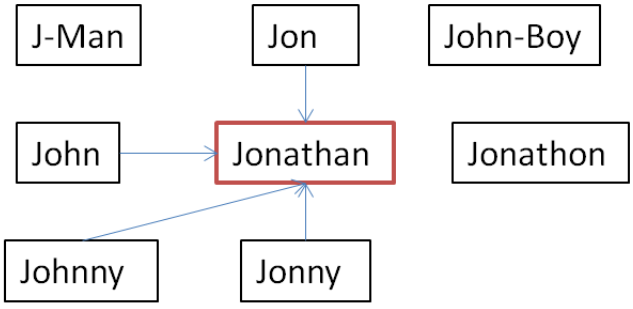
These rules contain a number of inherent conflicts. This is illustrated in the following diagram in which JONATHAN is identified as the Primary:



The arrows indicate the following:

Arrow Type	Reason for Conflict
	N/A (No conflict exists)
	PIV
	PVIM
	PVOM

The conflict resolution rules will discard the mappings that cause conflicts, as follows:



Resulting in the following mappings being created:

Name	Primary
JOHN	JONATHAN
JON	JONATHAN

Name	Primary
JONNY	JONATHAN
JOHNNY	JONATHAN

3

Using Matching

This chapter describes how you can use the EDQ-CDS matching functionality to match your data.

This chapter includes the following sections:

EDQ-CDS has been designed to match customer data that exhibits real-world variability. All relevant matches in the data set are presented back and appropriately scored according to the likelihood of a match between records. To do this, it uses a variety of different mechanisms, including the application of a wide range of matching algorithms on the data as it is presented, as well as matching techniques on derived forms of the data.

For example, names presented in one writing system are matched both using this writing system and also using a transformed version of the name, providing effective cross-script matching. Similarly, addresses are matched in near raw form (after standardization of international address words and phrases, and after removal of filler words), but also by extracting and matching key information from the address, such as the likely building number, sub-building number, and postal code.

3.1 Objectives of Matching

In general, the matching services provided by EDQ-CDS are designed for duplicate prevention, rather than searching. This means that the intention of the out-of-the-box services is to intervene when a record is added to a system if it appears that it may already exist. The implication of this is that the matching services are focused on much more than a single attribute (such as Name) and deliberately do not cast as wide a net as a typical search operation. There may be other records in the system that are not matched but which have similar details, perhaps even exactly the same name, but where the secondary identification information indicates that a match is unlikely. In these cases, EDQ-CDS aims to minimize the additional work for users or data stewards whose role it is to resolve possible matches. This makes the product ideally suited to operate as the data quality protection component of a Master Data

Management system, such as Oracle Customer Hub, where the purpose of the services is to link as many records as possible together automatically with as little noise as possible. The same is true for a Customer Relationship Management system, such as Siebel.

Note : It is possible to change the configuration of EDQ-CDS in order to perform more exhaustive matching. This is mainly designed for use with low volume, high value data sets that do not necessarily offer sufficient secondary information (beyond name fields).

3.1.1 Multiple Locales and Languages

EDQ-CDS has been designed as a multi-locale system, and uses international and culture-sensitive name transcription, transliteration and variant recognition techniques, as well as using international dictionaries when standardizing and matching addresses.

The system is designed to work with international data, and provides international dictionaries of name and address standardizations for this purpose. The international 'Latin script' dictionaries provide coverage of the following 'base' locales, amongst others:

- United States and Canada
- United Kingdom
- France
- Germany
- Italy
- Spain
- Portugal
- Brazil
- Greece
- Ireland
- Austria
- Turkey
- South Africa
- Australia and New Zealand
- Scandinavia
- Argentina
- Mexico

In addition to these base locales, EDQ-CDS provides specific optional capabilities for advanced handling of data from the following locales:

- Arab World (Arabic and Mixed Arabic/Latin)
- Japan (Kanji, Katakana and Hiragana)
- China (Simplified and Traditional Chinese)

- Russia
- Korea (Hangul)

The set of enabled languages is determined by the configuration of the EDQ-CDS - Initialize Reference Data project, so that the same reference data may be used by any number of EDQ-CDS matching servers. By default, reference data sets for the base locales are pre-initialized in the EDQ server landing area, but these can be easily overwritten either by unzipping `cdslists-initialized-full.zip` over these files (to provided coverage for all supported locales and languages) or by configuring and running the Initialization job.

3.1.2 Uses of Matching

The Matching processes included in EDQ-CDS are designed primarily for the following use cases:

- Duplicate Prevention - uses the Key Generation and Matching web services to prevent duplicate records being entered into applications.
- Regular Batch Matching for Duplicate Removal - uses the Batch Matching job, run on all, or a subset of, data in an application, and links records together for potential merge.

It is also possible to use the Batch Matching processes as a template for the deduplication of records before they are loaded into a system. This is likely to require additional configuration, and use of EDQ. In such circumstances the best practice is to understand the data before matching using data profiling and audit techniques, such as those available in the EDQ-CDS Data Quality Health Check. In most cases, the set of enabled match rules will need some tuning towards the specifics of the in-scope data in order to provide the optimum balance between performance and effectiveness. It may also be necessary to use EDQ's Match Review application to review possible matches, and construct rules for merging records together.

Note : EDQ-CDS does not provide any out-of-the-box merging (or survivorship) configuration, because in the two main use cases, merging is performed by the calling application after matches have been identified.

3.1.2.1 Duplicate Prevention

EDQ-CDS uses stateless web services for duplicate prevention to avoid complex replication and synchronization of large volume customer data. This places the following requirements on the application integrating with EDQ:

1. Storage of Cluster Key tables for each type of record (for example, Contacts or Accounts). These are normally thin tables with two columns - the Primary Key of the record and the Cluster Key. The table must allow for multiple key values per record.
2. Functionality to select and construct candidate records to submit to the Matching service. This involves:
    a. Querying the Cluster Key table for the relevant record, and finding all records that share a key value with the driving record.
    b. Constructing the data that is required for matching for each of these records.
    c. Submitting these Candidate records together with the driving record to the Matching service.

Optimum Duplicate Prevention Process Flow

In order to access the full capabilities of EDQ-CDS for duplicate prevention, the integration should work as follows:

1. To prepare the system for real-time duplicate prevention, key values are generated for each record in Batch using the Key Generation process. This can occur either when migrating the data into the application, or as a batch process to generate the key values into the application's Cluster Key tables.
2. When a record is added or updated in the application, the Key Generation service is called in real-time, and returns a number of key values for the record.
3. The application then selects candidate records (those records which share a common key with the driving record) using the existing stored keys and submits them along with the driving record to the Matching service.
4. The Matching service decides which of the candidates are a likely match to the driving record and returns the ids of these records, and a score indicating the strength of match.
5. The application then decides how to consume the matching results; for example, whether to 'auto-match' or present possible matches to the user so that a decision can be made whether or not to continue with inserting a record, or merge it with an existing record.
6. If the record is merged with another record to create a changed master record, an additional call should be made to the Key Generation service in order to re-generate the correct key values before committing the record.

In this model, complex multi-locale EDQ techniques are used to generate the key values and ensure that the right balance between performance and matching effectiveness is maintained, while ensuring that the calling application retains control of data integrity and transactional commits.

3.1.2.2 Batch Matching

When working with Siebel CRM, Siebel's Data Quality Manager is used to instigate batch jobs and a shared staging database is used to write records for matching and to consume match results. The EDQ-CDS batch matching processes automatically adjust to Siebel's 'Full Match' (match all records against each other) and 'Incremental Match' (match a subset of records against all of their selected candidates) modes.

3.1.3 Match Tuning

In EDQ-CDS matching, it is not necessary to be overly concerned with which identifiers will be populated in the data that is worked with. EDQ-CDS does not use an algorithm that will place unnecessary emphasis on unpopulated data, and so does not require adjustment for this.

Matching works by considering matches on related input attributes separately (such as those relating to name, address, email, etc) and attempting various ways on each of these to find a match. EDQ refers to the grouped matching rules on these logically related attributes as "compound comparisons". It then combines the matches on these compound comparisons to decide how well two records match as a whole. The matching design builds in the knowledge of how strong an identifier is likely to be based on real world principles. Match tuning is normally a matter of performing one of the following tasks:

- Adjusting the weighting of a compound comparison
- Enabling or disabling a compound comparison

- Adjusting the key generation configuration.
- Enabling or disabling a provided rule
- Adjusting the score for a specific rule within a compound comparison
- Inserting a new rule into a compound comparison (perhaps a stronger or weaker version of an existing rule)

> **Note:**
>
> Even when inserting new rule configurations, it may well be possible to use existing comparisons and comparison results rather than adding new comparisons, though both are possible.

3.1.4 Output of Match Metadata

The output of match metadata provides granular details about why two records matched, providing information about which compound comparisons contributed to a match. The following EDQ match metadata is output for each compound comparison (for example, Name, Address, Email, Phone, etc.):

- [Compound Comparison] Result, for example, `N040 Given name abbreviated`
- Score (out of 100)
- Category (Exact, Fuzzy, No data or Conflict)

3.2 Using Key Generation

Key Generation is used to minimize the work that is performed during the final stage of matching. It works by splitting the records into tranches (clusters), based on similarities in significant data fields. Only subsets of the data which share similar characteristics (and will therefore be placed in the same cluster) will be compared on a record-by-record basis during matching.

If loose clusters are used, there will be a large number of records in each cluster. This means that there is a reduced risk that true matches will be missed, but also that a greater amount of processing will be required to compare all the key generated records. A tighter key generation strategy will result in smaller groups and hence a reduced processing time, but will increase the likelihood that some true matches will not be detected.

EDQ-CDS is supplied with a number of different key method algorithms for individual, entity, and address data that use different combinations of key data fields in their construction. Each key method algorithm has been assigned a unique prefix code for easy identification, and to ensure keys from different key methods are not identical.

3.2.1 Legacy Clustering

Prior to release 12.2.1 key generation was referred to as clustering and the functionality provided was a much more restricted version of current key generation, although the principles were the same. Only three methods of "clustering" were provided with no easy scope for customization.

These "legacy" methods can still be used by setting the following in the run profile:

```
phase.*.process.*.uselegacykeygen = Y
```

and the levels set using

```
phase.Individual\ Keygen.process.*.clusterlevel = [1/2/3]
```

3.2.2 Structure of Key Methods

For each party type, key methods are grouped into key groups and key types.

For example, the individual 'Name Phone' key group contains all the key methods constructed using a combination of the name and phone attributes. Within this group, there are two key types:

FNMGNMPNR: key methods based on family name metaphone, given name metaphone and phone number rightmost characters

FNMPNL: key methods based on family name metaphone and phone number leftmost characters

Each key type then consists of one or more actual key methods, each one using varying lengths of the metaphone or leftmost/rightmost characters.

For example, the FNMPNL key type contains the following key methods:

`FNM4PNL6`: family name metaphone first 4, phone number last 6

`FNM4PNL7`: family name metaphone first 4, phone number last 7

`FNM4PNL8`: family name metaphone first 4, phone number last 8

These are categorized as 'strict', 'typical' and 'loose' respectively, as the length of the phone number substring used becomes larger and therefore provides a tighter key.

Key values generated using the last of these methods would take the form:

```
FNM4PNL8^MN^65065421
```

An automatic or 'encoded' key profile consists of a pipe-delimited list of key methods with their associated key priorities, for example:

```
AD112FNL5GNL5^10|GNW1FNL0^11|AD17AD25CTL10^12|FNM4PNL8^13|PNR6^14
```

Note that the key priorities are only relative within a specific profile, they have no intrinsic meaning.

3.2.2.1 Keys for Custom Attributes

Keys for custom attributes can optionally be created during Key Generation (by default keys are not generated for custom attributes).

This is specified in the run profile as follows:

```
phase.*.process.*.customstringNkey = Y

phase.*.process.*.customdateNkey   = Y
```

and can be overridden in real-time on a per-message basis as follows:

```
<dn:request customstringNkey="Y" customdateNkey="Y">
```

The actual keying method used depends on the key profile specified:

- Strict profiles key custom strings on the full string, and custom dates on the full date
- Loose profiles key custom strings on the metaphone of the string, and custom dates on the year only
- Typical profiles key custom strings on the first 10 characters of the string, and custom dates on the year and month

Note that custom attributes are ignored if legacy cluster levels are used.

3.2.3 Key Method Analysis

Key method analysis introduces the capability within CDS to automatically analyze the customer's data and determine the best key profile for that particular data set. Key analysis consists of these main steps:

1. Generate key values for the data using all available key methods.
2. Profile, score and rank those key values using various statistical mechanisms such as high frequency key values and distribution/diversity of key values.
3. Construct and output a recommended key profile by selecting the best key method(s) within each key group.

Custom attributes will be taken account of during Key Analysis if they are enabled for Key Generation, as described in Keys for Custom Attributes.

All available custom attribute key methods are analyzed, in a similar fashion to the existing fixed attributes.

3.2.3.1 Running Batch Key Analysis

There are several new staging tables for Key Analysis which must be created prior to running the job. The SQL commands to create these tables are added to the existing default script, `edq_staging_tables.sql`, which is delivered with EDQ and installed under `<middleware_home>/edq/oracle.edq/scripts/cds`.

The batch jobs for running Key Analysis are:

- Batch Individual Key Analysis
- Batch Entity Key Analysis
- Batch Address Key Analysis

These jobs are structured in a similar fashion to the existing batch jobs for key generation and matching, in that they expect to receive party data in the relevant Candidates tables in the staging schema, and output their results to tables in the same schema.

Note that due to the statistical nature of the way Key Analysis works, it expects to always receive the full set of customer data to analyze. Whilst the jobs will actually run with a sample of the data, the results will only apply to that sample and cannot be scaled up to the full dataset.

The following run profile parameter must be set to `Y` in order for Key Analysis to run successfully:

```
phase.Key\ Analysis.process.*.generateallkeys = Y
```

Note also that the run profile contains various new SQL statements for Key Analysis, in order to expose the `SERVERID` and `JOBID` parameters in a similar fashion to the existing staging tables. Therefore these parameters will also need updating in the run profile in-line with any changes for the other table parameters.

3.2.3.2 Key Method Analysis Outputs

Key Analysis outputs results in the following staging tables:

3.2.3.2.1 EDQCDS_KEY_ANALYSIS_PROFILE This table contains a single row per job, which simply contains the recommended key profile, consisting of a pipe-delimited list of key methods with their associated key priorities, for example:

```
AD112FNL5GNL5^10|GNW1FNL0^11|AD17AD25CTL10^12|FNM4PNL8^13|PNR6^14
```

This is the profile that should be used for key generation and matching, if the user decides to accept the recommendation.

> **Note:** Key Analysis does not actually output the key values for the recommended profile; this must be done separately by running the relevant batch key generation job and passing in the recommended profile accordingly.

3.2.3.2.2 EDQCDS_KEY_ANALYSIS_REPORT This table contains a single row per key method analyzed, detailing the statistics and score for each method together with an indication of whether it was selected for the profile and if so the assigned priority. Only those key methods generated are listed - that is, those for which the party data contained relevant non-blank attributes.

This report is provided mainly for support and diagnostics purposes.

3.2.3.2.3 EDQCDS_KEY_ANALYSIS_TOP_VALUES This table contains the top 20 key values by count per key method analyzed. Only those key methods generated are listed - that is, those for which the party data contained relevant non-blank attributes.

This report can help users identify potential DQ issues with their data, i.e. very large key value counts may indicate spikes and generic data values such as '000000' phone numbers or 'sales@' email addresses.

3.2.4 Individual Key Types

Key methods for matching individual data are based on the following key types:

Table 3–1 Address Only

Key Type	Description
AD1AD2CTL	address1 distilled (No whitespace, leftmost chars), address2 distilled (No whitespace, leftmost chars), city standardized (No whitespace, leftmost chars)
ADACTLPRE	adminarea standardized (No whitespace, leftmost chars), city standardized (No whitespace, leftmost chars), premise derived (Denoised, no whitespace, leftmost chars)

Table 3–2 Name and Company

Key Type	Description
ANLGNLFNL	accountname (No whitespace, leftmost chars), givenname standardized (No whitespace, leftmost chars), familyname (No whitespace, leftmost chars)
ANWFNMGNL	accountname (Leftmost words), familyname (Double metaphone, leftmost chars), givenname standardized (No whitespace, leftmost chars)
ANWFNM	accountname (Leftmost words), familyname (Double metaphone, leftmost chars),

Table 3–2 (Cont.) Name and Company

Key Type	Description
ANMGNLFNL	accountname (First word, double metaphone, leftmost chars), givenname standardized (No whitespace, leftmost chars), familyname (No whitespace, leftmost chars)

Table 3–3 Name and DOB

Key Type	Description
DBYGNLFNL	DOB standardized (Year), givenname standardized (No whitespace, leftmost chars), familyname (No whitespace, leftmost chars)
DBXGNLFNL	DOB standardized (Full date), givenname standardized (No whitespace, leftmost chars), familyname (No whitespace, leftmost chars)
DBNGNLFNL	DOB standardized (Year and month), givenname standardized (No whitespace, leftmost chars), familyname (No whitespace, leftmost chars)

Table 3–4 Name Only

Key Type	Description
FMP	fullname standardized (Array of tokens, metaphone pairs, leftmost chars)
GNWFNL	givenname standardized (Leftmost words), familynamenormalized (No whitespace, leftmost chars)

Table 3–5 Name and Phone

Key Type	Description
FNMGNMPNR	familyname (Double metaphone, leftmost chars), givenname standardized (First word, double metaphone, leftmost chars), phonenumbers standardized (Rightmost chars (array of))
FNMPNL	familyname (Double metaphone, leftmost chars), phonenumbers standardized (Leftmost chars (array of))

Table 3–6 Full Name and Address

Key Type	Description
AD1FNLGNL	address1 distilled (No whitespace, leftmost chars), familyname (No whitespace, leftmost chars), givenname standardized (No whitespace, leftmost chars)
FNLGNLPCL	familyname (No whitespace, leftmost chars), givenname standardized (No whitespace, leftmost chars), postalcode standardized (No whitespace, leftmost chars)
CTLFNLGNL	city standardized (No whitespace, leftmost chars), familyname (No whitespace, leftmost chars), givenname standardized (No whitespace, leftmost chars)

Table 3–7 Household Address

Key Type	Description
AD1FNMPCL	address1 distilled (No whitespace, leftmost chars), familyname (Double metaphone, leftmost chars), postalcode standardized (No whitespace, leftmost chars)
AD1FNMCTL	address1 distilled (No whitespace, leftmost chars), familyname (Double metaphone, leftmost chars), city standardized (No whitespace, leftmost chars)

Table 3–8 National ID

Key Type	Description
NIL	nationalidnumber standardized (Leftmost chars (array of))
NIP	nationalidnumber standardized (Pairs of leftmost & rightmost chars (array of))

Table 3–9 Phone

Key Type	Description
PNR	phonenumbers standardized (Rightmost chars (array of))

Table 3–10 Script Name

Key Type	Description
OSLPCL	scriptfullname (No whitespace, leftmost chars), postalcode standardized (No whitespace, leftmost chars), ()

Table 3–11 Tax Number

Key Type	Description
TNL	taxnumber standardized (Leftmost chars (array of))
TNP	taxnumber standardized (Pairs of leftmost & rightmost chars (array of))

Table 3–12 UID

Key Type	Description
UID(1/2/3)	uid[1, 2, 3]standardized (Leftmost chars (array of))

Table 3–13 Custom Strings

Key Type	Description
CM[1-6]	customstring[1-6] standardized (Double metaphone, leftmost chars, if blank 8 leftmost chars (no metaphone))
CL[1-6]	customstring[1-6] standardized (No whitespace, leftmost chars)

Table 3–14 Custom Dates

Key Type	Description
CY[1-6]	customdate[1-6]standardized (Year)

Table 3–14 (Cont.) Custom Dates

Key Type	Description
CX[1-6]	customdate[1-6] standardized (Full date)
CN[1-6]	customdate[1-6] standardized (Year and month)

Note : The key method algorithms use data attributes that have been normalized (for example, converted to upper case and symbols stripped) and have had whitespace removed. This allows key generation and matching to be performed in a case-insensitive manner and to be tolerant of the spacing within attributes.

3.2.4.1 Examples

The following record data is used to provide examples of the key values that are generated by the individual key method algorithms:

Attribute	Value
firstname	Jim
middlename	Frederick
lastname	Smith
mobilephone	077777 123456
email	jsmith@mymail.com
taxnumber	888666444
accountname	Acme Ltd
address1	14 high St
city	Cambridge
postalcode	CB1 2AB
uid1	00021-53563
eid1	gbr0008873323
nationalidnumber	AB 12 34 56 C

The key values that are generated using the Typical key profile are as follows:

Key Type	Key Method	Priority	Cluster Values
UI1	UI10	1	UI10^0002153563
AD1FNLGNL	AD110FNL3GNL3	42	AD110FNL3GNL3^14HIGH^SMI^JAM
AD1FNMPCL	AD12FNM3PCL5	55	AD12FNM3PCL5^14^SM0^CB12A
AD1AD2CTL	AD17AD25CTL5	59	AD17AD25CTL5^14HIGH^^CAMBR
ANWFNM	ANW1FNM4	54	ANW1FNM4^ACME^SM0
CTLFNLGNL	CTL10FNL3GNL3	51	CTL10FNL3GNL3^CAMBRIDGE^SMI^JAM

Key Type	Key Method	Priority	Cluster Values
ENP	ENP15	40	ENP15^JSMITHMYMAILCOM
FNLGNLPCL	FNL3GNL1PCL5	44	FNL3GNL1PCL5^SMI^J^CB12A
FNMPNL	FNM4PNL7	46	FNM4PNL7^SM0^0777771
NIL	NIL10	36	NIL10^AB123456C
PNR	PNR6	47	PNR6^123456
TNL	TNL1	37	TNL10^888666444

3.2.5 Entity Key Types

The following key types are provided for matching entity data:

Table 3–15 Name Address

Key Type	Description
AD1EMTPCL	address1 distilled (No whitespace, leftmost chars), entityname distilled (Array of tokens, double metaphone, leftmost chars), postalcode standardized (No whitespace, leftmost chars)
ENLPCL	entityname distilled/normalized (No whitespace, leftmost chars), postalcode standardized (No whitespace, leftmost chars),
FANENLCTL	fulladdress distilled (No whitespace, no numbers, denoised, leftmost chars), entityname distilled/normalized (No whitespace, leftmost chars), city standardized (No whitespace, leftmost chars)
AD1ENLPCL	address1 distilled (No whitespace, leftmost chars), entityname distilled/normalized (No whitespace, leftmost chars), postalcode standardized (No whitespace, leftmost chars)

Table 3–16 Name Metaphone Address

Key Type	Description
CTLFALNSM	city standardized (No whitespace, leftmost chars), fulladdress distilled (No whitespace, leftmost chars), fullname distilled/normalized (Double metaphone, leftmost chars)
FALNSM	fulladdress distilled (No whitespace, leftmost chars), fullnamedistilled/normalized (Double metaphone, leftmost chars),
CTLNSM	city standardized (No whitespace, leftmost chars), fullname distilled/normalized (Double metaphone, leftmost chars),

Table 3–17 Name only

Key Type	Description
NSL	fullname distilled (No whitespace, leftmost chars)
ENMSNM	entityname distilled (Double metaphone, leftmost chars), entitysubname distilled (Double metaphone, leftmost chars)
FMT	fullname distilled (Array of tokens, double metaphone, leftmost chars)

Table 3–18 Name City Phone

Key Type	Description
CTLENLPNR	city standardized (No whitespace, leftmost chars), entityname distilled/normalized (No whitespace, leftmost chars), phonenumbers standardized (Rightmost chars (array of))
CTLENLPNL	city standardized (No whitespace, leftmost chars), entityname distilled/normalized (No whitespace, leftmost chars), phonenumbers standardized (Leftmost chars (array of))

Table 3–19 Phone

Key Type	Description
PNR	phonenumbers standardized (Rightmost chars (array of))

Table 3–20 Website

Key Type	Description
WSL	websitestem (Leftmost chars (array of))

Table 3–21 Script Name

Key Type	Description
OSL	script fullname (Array of tokens, leftmost chars)

Table 3–22 VAT number

Key Type	Description
VNL	vatnumber standardized (Leftmost chars (array of))
VNP	vatnumber standardized (Pairs of leftmost & rightmost chars (array of))

Table 3–23 Tax Number

Key Type	Description
TNL	taxnumber standardized (Leftmost chars (array of))
TNP	taxnumber standardized (Pairs of leftmost & rightmost chars (array of))

Table 3–24 UID

Key Type	Description
UID[1,2,3]	uid[1, 2, 3]standardized (Leftmost chars (array of))

Table 3–25 Custom Strings

Key Type	Description
CM[1-6]	customstring[1-6] standardized (Double metaphone, leftmost chars, if blank 8 leftmost chars (no metaphone))
CL[1-6]	customstring[1-6] standardized (No whitespace, leftmost chars)

Table 3–26 Custom Dates

Key Type	Description
CY[1-6]	customdate[1-6]standardized (Year)
CX[1-6]	customdate[1-6] standardized (Full date)
CN[1-6]	customdate[1-6] standardized (Year and month)

Note : The key method algorithms use data attributes that have been normalized (for example, converted to upper case and symbols stripped) and whitespace removed. This allows key generation and matching to be performed in a case-insensitive manner and be tolerant to the spacing within attributes.

3.2.5.1 Examples

The following record data is used to provide examples of the key values that are generated by the entity key method algorithms:

Attribute	Value
name	Oracle UK
subname	Cambridge
phone	+441223228400
website	http://www.oracle.com/uk
taxnumber	RGW432D243224
vatnumber	999111
address1	296 Cambridge Science Park
city	Cambridge
postalcode	CB4 0WD
uid1	00021-53563
eid1	gbr0008873323

The following key values are generated using a key profile of Typical:

Key Type	Key Method	Priority	Key values
AD1PCL	AD13PCL4	43	AD13PCL4^296^CB40
AD1EMTPCL	AD14EMT4PCL3	41	AD14EMT4PCL3^296C^ARKL^CB4
CTLNSM	CTL0NSM6	49	CTL0NSM6^CAMBRIDGE^ARKLKM
CTLENLPNL	CTL1ENL1PNL7	47	CTL1ENL1PNL7^C^O^4412232
ENLPCL	ENL4PCL3	42	ENL4PCL3^ORAC^CB4
FALNSM	FAL10NSM4	39	FAL10NSM4^296CAMBRID^ARKL NSL25^ORACLECAMBRIDGE
NSL	NSL25	40	NSL25^ORACLECAMBRIDGE
PNR	PNR6	58	PNR6^228400

Key Type	Key Method	Priority	Key values
TNL	TNL10	35	TNL10^RGW432D243
UI1	UI10	1	UI10^0002153563
VNL	VNL10	36	VNL10^999111
WSL	WSL8	57	WSL8^ORACLE

3.2.6 Address Key Types

The following key method types are provided for matching address data:

Table 3–27 Address Lines

Key Type	Description
AD1AD2	address 1 distilled (No whitespace, leftmost chars), address 2 distilled (No whitespace, leftmost chars)

Table 3–28 Address City

Key Type	Description
AD1CTL	address 1 distilled (No whitespace, leftmost chars), citystandardized (No whitespace, leftmost chars)
CTLPCLPRE	citystandardized (No whitespace, leftmost chars), postalcodestandardized (No whitespace, leftmost chars), premisederived (Denoised, no whitespace, leftmost chars)
PMSPCC	premisederived/address 1 distilled (First number word of premisederived/ premise leftmost chars/first number word of address1distilled/left most chars address1 distilled), postalcodestandardized/citystandardized (Leftmost chars of postalcode standardized/leftmost chars of city standardized),

Table 3–29 Full Address

Key Type	Description
FAL	fulladdress distilled (No whitespace, leftmost chars)
FAN	fulladdress distilled (No whitespace, no numbers, denoised, leftmost chars)

Table 3–30 Postal Code

Key Type	Description
PCL	postalcode standardized (No whitespace, leftmost chars)

Note:

- A **Number word** is a word with one or more numbers within it. for example, 234 and 2A are both number words.
- The key method algorithms use data attributes that have been normalized (for example, converted to upper case and symbols stripped) and whitespace removed. This allows key generation and matching to be performed in a case-insensitive manner and be tolerant to the spacing within attributes.

3.2.6.1 Examples

The following record data is used to provide examples of the key values that are generated by the address key method algorithms:

Attribute	Value
address1	2529 CINCINNATI ST
address2	APT 6
city	LOS ANGELES
adminarea	CA
postalcode	90033

Note : During Key generation, ST is distilled out of the address1 field, and APT is distilled out of the address2 field. This is because they are common addressing components that are less important identifiers than the remainder of the address line, and removing them produces more accurate clusters.

The Key values that are generated using the Typical address key profile are:

Key Type	Key Method	Priority	Key Values
AD1AD2	AD110AD210	12	AD110AD210^2529CINCIN^6
AD1CTL	AD15CTL8	9	AD15CTL8^2529C^LOSANGEL
CTLPCLPRE	CTL8PCL5PRE0	10	CTL8PCL5PRE0^LOSANGEL ^90033^2529
FAL	FAL10	11	FAL10^2529CINCIN
FAN	FAN10	13	FAN10^CINCINNATI
PCL	PCL0	15	PCL0^90033
PMSPCC	PMS6PCC5	8	PMS6PCC5^2529^90033

3.3 Using Individual Matching

The matching design for individuals in CDS is based on combining matches between several logical identifiers (compound comparisons). These compound comparisons are:

- Name

- Address
- Account name
- DOB
- Phone number
- Email
- National ID number
- Tax number

It is also possible to enable matching of the custom fields (however, these are not enabled by default)

EDQ-CDS uses preconfigured match rules on the compound comparisons to ascertain how well two records match (or don't match) on that particular logical identifier.

In order to determine whether two records as a whole match, EDQ-CDS uses the results for the matching on the logical identifiers and combines them to produce an overall score that gives a measure of how well the records match. Note that a conflict will negatively affect a score, as well as a match increasing it. For example, two records with an exact match on name and address, but a conflicting date of birth will score lower than a two records with an exact match on name and address, but no date of birth.

Each logical identifier has a default weighting, defining how likely two records with matches on the compound comparison related to this logical identifier.

3.3.1 Matching on the Individual Name logical identifier

The rules for matching on the individual name compound comparisons include the use of pre-matching transformations and various matching comparisons in order to handle the following types of variance between different representations of what may be the same individual name:

- Names written in different writing systems/scripts, for example, '?????' and 'Zoran'.
- Variants of the same name, for example, 'Bill' and 'William'.
- Different levels of name completeness, for example, 'Joseph Andrew Harris' and 'Joseph Harris'.
- Name tokens in a different order, for example, 'Lacazette Jacques' and 'Jacques Lacazette'.
- Abbreviated forms of names, for example, 'Chris' and 'Christian'.
- Typographic differences, for example, 'Michael' and 'Micheal'.
- The use of initials, for example, 'A M' and 'Alexander Martin'.
- Changes of surname due to marriage, for example, 'Paula Jones' and 'Paula Lewis' at the same address.
- Various combinations of the above types of variance.

Note : In this table the pipe character is used to indicate a separator between the input given name and family name attributes (for example, Given Name= Martin, Family Name=Smith is written as 'Martin | Smith'). Where no pipe character is used, this means the Full Name is used in the match rule.

Note Also: Near the top of this list are some conflict name rules, these are designed to negatively weight matches between two names that are obviously different genders, to avoid matches of this type.

Name Matching Rules	Example Name Match	Type
Script full name exact	Зоран Александрович Макаров =Зоран Александрович Макаров	Exact
Name exact	Martin | Fox = Martin | Fox	Exact
Standardized given name	Bill | Lewis = William | Lewis	Exact
Given name abbreviated	Chris | Smith = Christina | Smith	Fuzzy
Name conflict, supplied gender different	Paula | Smith - Paul Smith (negatively weighted to eliminate matches such as this)	Conflict
Name conflict, derived gender different	Paula | Smith - Paul Smith (negatively weighted to eliminate matches such as this)	Conflict
Standardized given name abbreviated	Abell | Hernandez = Abelson | Hernandez	Fuzzy
Script full name any order	Макаров Зоран Александрович =Зоран Александрович Макаров	Fuzzy
Given name similar and sounds like	Yngrid | Martin = Ingrid | Martin	Fuzzy
First name similar and sounds like	Yngrid Elisabeth | Martin = Ingrid Martin	Fuzzy
Additional given names	Michael John | Smith = John | Smith	Fuzzy
Standardized full name	Mehmood Mahomed = Mahmoud Mohammed	Fuzzy
Script full name has additional names	Зоран Макаров =Зоран Александрович Макаров	Fuzzy
Additional names	Mary Jones Steward = Mary Jones	Fuzzy
Script full name typos	Зоран Александрович Макаров =Зоран Александрович Маккаров	Fuzzy
Standardized given name abbreviated; family name typos	Abell | Hernandez = Abelson | Hernandes	Fuzzy
Full name typos, all words	Mary Cloire Jonez = Mary Claire Jones	Fuzzy
First name first three; family name typos	Ros Susan | Jonez = Rose Susan | Jones	Fuzzy
Full name initials in order; additional names	G A | Smith = Gordon Alfred | Smith	Fuzzy

Name Matching Rules	Example Name Match	Type
Standardized first name only; female	Jacklin|Jones = Jacqueline|Smith	Fuzzy

3.3.2 Matching on the other logical identifiers

Addresses

The rules for matching on the address compound comparison within individual name matching include the use of pre-matching transformations and various matching comparisons in order to handle the following types of variance between different representations of what may be the same address:

- Extracting the premise and subpremise
- Standardizing commonly used words such as STREET, ROAD, etc.
- Stripping commonly used words such as STREET, ROAD, etc.
- Typographic differences

> **Note:** In this table the pipe character is used to indicate a separator between the inputs of address1, address2, address3, city, adminarea, postalcode. For example address1=296 Cambridge Science Park address2= Milton Road address3=<blank>, city=Cambridge adminarea = <blank> postalcode=CB4 0WD is represented as 296 Cambridge Science Park|Milton Road|||Cambridge|||CB4 0WD

Table 3–31 Matching on other logical identifiers

Address Rule Name	Example	Type
Address exact	296 Cambridge Science Park|Milton Road||Cambridge||CB4 0WD = 296 Cambridge Science Park|Milton Road||Cambridge||CB4 0WD	Exact
Premise, subpremise, address similar, postal code	Flat 1|296 Cambridge Science Park||Cambridge||CB4 0WD = Flat 1|296 Cambridge Sci Park||Cambridge||CB4 0WD	Fuzzy
Premise, no subpremise, address similar, postal code	296 Cambridge Science Park|Milton Road||Cambridge||CB4 0WD = 296 Cambridge Sci Park|Milton Road||Cambridge||CB4 0WD	Fuzzy
Address1 and address2 distilled exact, postal code starts with	296 Milton Road|||Cambridge||CB4 0WD = 296 Milton Road||||CB4 0WD	Fuzzy
Address1 distilled exact, address2 no conflict, postal code starts with	296 Milton Road|Science Park||Cambridge||CB4 0WD = 296 Milton Road||||CB4 0WD	Fuzzy
Premise, subpremise, postal code starts with	Flat 1|352 Milton Road||Cambridge||CB4 0WD = 352 Milton Road|Flat 1|||CB4 0WD	Fuzzy
Premise, no subpremise, postal code starts with	296 Cambridge Science Park|||Cambridge||CB4 0Wd = 296 The Science Park||||CB4 0WD	Fuzzy
Address1 distilled exact, postal code starts with	296 Cambridge Science Park|Flat 1||Cambridge||CB4 0WD = 296 Cambridge Science Park|Flat 6||Cambridge||CB4 0WD	Fuzzy

Table 3–31 (Cont.) Matching on other logical identifiers

Address Rule Name	Example	Type
Address all words	296 Science Park|Milton Road||Cambridge||CB4 0WD = Science Park|Milton Road||||CB4 0WD	Fuzzy
Address all words typos	296 Science Park|Milton Road||Cambridge||CB4 0WD = Sciense Park|Milton Road||||CB4 0WD	Fuzzy
Address similar, postal code	296 Science Pk|Milton Rd||Cambridge||CB4 0WD = Sceince Park|Milton Road||Cmbridge||CB4 0WD	Fuzzy
Address similar; first address1 word	297 Cambridge Science Park||Milton Road|||CB30WS = 296 Cambridge Science Park|Milton Road||||CB4 0WD	Fuzzy
Postal code	296 Science Park||||CB4 0WD = |Milton Road||||CB4 0WD	Fuzzy
Postal code starts with	296 Science Park||||CB4 0WD = |||||CB4	Fuzzy
City exact	352 Mill Road|||Cambridge||CB1 3NN = 296 Cambridge Science Park|Milton Road||Cambridge||CB4 0WD	Fuzzy
Address no data	||||| = 296 Cambridge Science Park|Milton Road||Cambridge||CB4 0WD	No data
Address conflict	19 Teme Ave|||Malvern|Worcs|WR14 2XA = 296 Cambridge Science Park|Milton Road||Cambridge||CB4 0WD	Conflict

Account name

Matching on account name allows for matches including

- Exact match
- Typographic differences
- All words in common

Table 3–32 Account name

Account name rule	Example	Type
Account name exact	Widgets and Gadgets Ltd = Widgets and Gadgets Ltd	Exact
Account name typos	Widgets and Gadgets Ltd = Widgets and Gagets Ltd	Fuzzy
Account name all words	Federal Mogul Camshafts Castings Ltd = Federal Mogul Camshafts Ltd	Fuzzy
Account name all words out of order	Federal Mogul Camshafts Castings Ltd = Federal Mogul Castings Camshafts Ltd	Fuzzy
Account name all words typos	Federal Mogul Camshafts Castings Ltd = Federal Mogul Camshfts Ltd	Fuzzy
Account name all words output of order typos	Federal Mogul Camshafts Castings Ltd = Federal Mogul Castings Camshfts Ltd	Fuzzy
Account name no data	Oracle Ltd =	No data
Account name conflict	Federal Mogul Camshafts Castings Ltd = Wigets and Gadgets Ltd	Conflict

Phone numbers

Table 3–33 Phone numbers

Phone matching rule	Example	Type
Phone exact	01223456678 = 01223456678	Exact
Phone last N	+44223456678 = 01223456678	Fuzzy
Phone no data	01223456678 =	No data
Phone conflict	01223456678=01684345678	Conflict

Email

Email matches allow for matches including:

- Exact match
- User name only exact
- Typographic errors

Table 3–34 Email

Email match rule	Example	Type
Email exact	someonesname@company.com = someonesname@company.com	Exact
Email user exact	someonesname@company.com = someonesname@adomain.com	Fuzzy
Email typos	someonesname@companion.com = someonesname@company.com	Fuzzy
Email no data	someonesname@company.com =	No data
Email conflict	someonesname@company.com = aperson@adomain.com	Conflict

Date of birth

Date of birth matches allow for matches including:

- Exact match
- Transposition of day/month match

Date of birth match rules also include a conflict rule where very different dates are penalized more severely

Table 3–35 Date of birth

Date of birth match rule	Example	Type
Date exact	11/01/1976 = 11/01/1976	Exact
Date similar	01/11/1976 = 11/01/1976	Fuzzy
Date no data	11/01/1976 =	No data
Date too different	11/12/2001 = 11/01/1976	Conflict
Date conflict	11/01/1976 = 20/01/1976	Conflict

National Id number

Table 3–36 National Id number

National Id number rule	Example	Type
National Id number exact	ABC112345 = ABC112345	Exact
National Id number typos	ABC12345 = ABC112345	Fuzzy
National Id number no data	ABC12345 =	No data
National Id number conflict	ABD2535 = BCD2145	Conflict

Tax number

Table 3–37 Tax number

Tax number rule	Example	Type
Tax number exact	ABC112345 = ABC112345	Exact
Tax number typos	ABC12345 = ABC112345	Fuzzy
Tax number no data	ABC12345 =	No data
Tax number conflict	ABD2535 = BCD2145	Conflict

The individual matching service outputs fields which give information on the matching of any of the logical identifiers described above, as well as an overall score and a overall rule name. This will allow the consuming application to have more granular information about how the records matched, to use as they wish.

Here is an example. The records in Table 3–38 were compared. Results are given in Table 3–39.

Table 3–38 Comparing Records

Record 1		Record 2	
Firstname	John	Firstname	J
Lastname	Smith	Lastname	Smith
Phonenumber	01223456789	Phonenumber	+44223456789
address1	35 Mill Road	address1	35 Mill Road
city	Cambridge	city	Cambridge
postalcode	CB1 2JJ	postalcode	CB1 2JJ

Table 3–39 Results of Comparison

Value	Result
matchscore	95
rulename	N040 Given name abbreviated, A010 Address exact, C070 Account name no data, D030 DOB no data, P020 Phone Last N, E040 Email no data, I030 National ID number no data, T030 Tax number no data
ruleattributes	NAME,ADDRESS,PHONE
comparisonresults	Name Fuzzy, Address Exact, Phone Fuzzy
namescore	95

Table 3–39 (Cont.) Results of Comparison

Value	Result
nameresult	N040 Given name abbreviated
namecategory	Fuzzy
addressscore	100
addressresult	A010 Address exact
phoneresult	P020 Phone Last N
phonescore	90
phonecategory	Fuzzyfamilyname

*Results that are no data are omitted for brevity

Note: If a field is known never to be populated in the data, then it is possible to "turn off" the compound comparison relating to the logical identifier, so that it does not appear in the rule.

The comparisonresults output field gives a comma separated list of any logical identifiers that have contributed to the match and the category of the match (i.e. returned a category of Exact or Fuzzy).

The ruleattributes field returns a comma separated list of the logical idenfiers that contributed to the match.

In addition to the logical identifiers described above, it is possible to configure individual matching to use Custom fields for matching. Custom fields are not enabled by default for either matching or clustering, for further information, see Section 3.6, "Using Matching with Customer-Added Attributes"

It is also possible to perform matching or elimination of Individual records using custom unique identifiers, see Section 3.5, "Using ID Matching."

3.4 Using Entity Matching

As with individuals, the matching design for entities in CDS is based on combining matches between several logical identifiers, using compound comparisons. These compound comparisons are:

- Entity name
- Address
- Phone number
- Website address
- Tax number
- VAT number

It is also possible to enable matching on the custom fields (however matching on these is not enabled by default)

EDQ-CDS uses preconfigured rules on the compound comparisons relating to the logical identifiers to ascertain how well two records match (or don't match) for that particular logical identifier.

In order to determine whether two records as a whole match, EDQ-CDS uses the results for the matching on the logical identifiers and combines them to produce an overall score that gives a measure of how well the records match. Note that a conflict will negatively affect a score, as well as a match increasing it. For example, two records with an exact match on name and address, but a conflicting web address will score lower than two records with an exact match on name and address, but no web address.

Each logical identifier has a default weighting, defining how likely two records with matches on this particular identifier are to be the same individual.

Note : It is significantly harder to match entities (as opposed to individuals) between different writing systems, as the process of transliteration — and even transcription — is much less likely to be successful. Very often, the only way to recognize that a company is the same when written in two different languages is to hold huge dictionaries of all possible company names and their appropriate translations (rather than transliterations or transcriptions). In most cases, such data is simply not available though if it is available it can be plugged into EDQ-CDS in order to improve results.

3.4.1 Entity Name Matching

The rules for matching entity names include the use of pre-matching transformations and various matching comparisons in order to handle the following types of variance between different representations of what may be the same entity name:

- Entity names written in different writing systems.
- Entity names with or without suffixes, for example, 'Oracle LTD' and 'Oracle'.
- Entity names containing abbreviated terms or suffixes, for example, 'Oracle Limited' and 'Oracle LTD'.
- Character order and spelling differences/errors in entity names, for example, 'Oracle' and 'Oralce'.
- Entity names with different levels of name completeness, for example, 'ABC Technology Consultants LTD' and 'ABC Technology LTD'.
- Entity name tokens appearing in a different order, for example, 'Cambridge Science Park LTD' and 'Science Park Cambridge'.
- Entity Names where part or all of the name is reduced to an acronym, for example, 'Oracle Catering' and 'O.C.'.

Note : In the following table, where a name matching rule uses the 'full name', this means it applies to the entity full name identifier, a concatenation of the entity name and sub-name attributes. The pipe (|) character is used to separate the entity name and sub-name were the sub-name attribute is required to provide an example match.

Entity Name Matching Rule	Example Entity Name Match	Type
Script full name exact	ДИРЕКЦИЯ БАЛТ-Й АЭС = ДИРЕКЦИЯ БАЛТ-Й АЭС	
Full name exact	TCHIBO GMBH = TCHIBO GMBH	
Standardized full name exact	ORACLE UK LTD | READING = ORACLE UK LIMITED | READING	Fuzzy
Script full name without suffixes exact	Открываем частное образовательное учреждение = Открываем частное образовательное	Fuzzy
Full name without suffixes exact	ORACLE = ORACLE CORPORATION	Fuzzy
Full name without suffixes similar and sounds like	ORACLE CAMBRIDGE SCIENCE PARK = ORACLE CAMBRIDGE PARK SCIENCE	Fuzzy
Script full name out of order	ДИРЕКЦИЯ БАЛТ-Й АЭС = ДИРЕКЦИЯ АЭС БАЛТ-Й	Fuzzy
Script full name without suffixes all words out of order	ОТКРЫВАЕМ ЧАСТНОЕ = ОТКРЫВАЕМ ОБРАЗОВАТЕЛЬНОЕ	Fuzzy
Full name without suffixes all words out of order	CAMBRIDGE SCIENCE PARK LTD = SCIENCE PARK CAMBRIDGE	Fuzzy
Script full name has additional names	Открываем частное учреждение | Москва = Открываем частное образовательное учреждение | Москва	Fuzzy
Script entity name without suffixes exact	ОТКРЫВАЕМ ЧАСТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ | МОСКВА = ОТКРЫВАЕМ ЧАСТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ | Колпино	Fuzzy
Entity name without suffixes exact	ORACLE CORPORATION | CAMBRIDGE = ORACLE | READING	Fuzzy
Full name all words shorter with typos	Oracle Inc | Cambridge =Oracl | Cambridge	Fuzzy
Script entity name without suffixes starts with	ОТКРЫВАЕМ ЧАСТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ | МОСКВА = ОТКРЫВАЕМ ЧАСТНОЕ УЧРЕЖДЕНИЕ | Колпино	Fuzzy
Entity name without suffixes starts with	ABC TECHNOLOGY CONSULTANTS LTD = ABC TECHNOLOGY LTD	Fuzzy
Script full name without suffixes all words shorter with typos	ОТКРЫВАЕМ ЧАСТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ = ОТКРЫВАЕМ ЧАСТНОЕБ	Fuzzy
Full name without suffixes all words shorter with typos	Federal Mogull | Camshafts Inc = Federal Mogul Camshafts Castings Ltd	Fuzzy
Script full name typos	ОТКРЫВАЕМ ЧАСТНОЕ УЧРЕЖДЕНИЕ | МОСКВА = ОТКРЫ ЧАСТНОЕ УЧРЕЖДЕНИЕ | МОСКВА	Fuzzy
Full name typos	ABD SERVICES LTD = ABC SERVICES LTD	Fuzzy
Script full name without suffixes typos	ОТКРЫВАЕМ ЧАСТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ = ОТКРЫВА ЧАСТНОЕ ОБРАЗОВАТЕЛЬНОЕ	Fuzzy
Full name without suffixes typos	ABD ENGINEERING LTD = ABC ENGINEERING	Fuzzy
Script entity name without suffixes starts with	ОТКРЫВАЕМ ЧАСТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ = УЧРЕЖДЕНИЕ ОТКРЫВАЕМ	Fuzzy
Entity name without suffixes starts with	ABC LIMITED| CAMBRIDGE = ABC PHARMACEUTICALS LIMITED| READING	Fuzzy
Standardized full name acronym exact	CSC= Computer Science Corporation	Fuzzy

Entity Name Matching Rule	Example Entity Name Match	Type
Entity name distilled longest common substring 12+	Colebrook & Burgess (North Shields) Ltd. = Colebrook & Burgess (Teesside) Ltd.	Fuzzy
Full name without suffixes acronym exact	CSC = Computer Science Collaborations Ltd	Fuzzy
Full name without suffixes acronym contains	Oracle CK = Oracle Collaborative Koopers	Fuzzy
Entity name without suffixes loose typos	Oracle Collaborative Coopers = Orracl Colabarativ Kupers	Fuzzy
Entity name without suffixes first token	DANVERS BANCORP INC = DANVERS MUNICIPAL FEDERAL CREDIT UNION	Fuzzy
Entity name distilled first 3 exact, longest common substring 6+	Lincoln Co-Operative Chemists Ltd. = Lincolnshire Co-Operative Ltd.	Fuzzy
Entity name distilled one or more tokens exact	Burgess Video Ltd. = Sue Burgess Ltd.	Fuzzy
Entity name no data	Oracle Corporation =	No data
Entity name conflict	Oracle Corporation = Sue Burgess Ltd.	Conflict

3.4.2 Matching on other logical identifiers for Entities

Addresses

The rules for matching addresses within entity name matching include the use of pre-matching transformations and various matching comparisons in order to handle the following types of variance between different representations of what may be the same address:

- Extracting the premise and subpremise
- Standardizing commonly used words such as STREET, ROAD, etc
- Stripping commonly used words such as STREET, ROAD, etc
- Typographic differences

> **Note:** In this table the pipe character is used to indicate a separator between the inputs of address1, address2, address3, city, adminarea, postalcode, country. For example address1=296 Cambridge Science Park address2= Milton Road address3=<blank>, city=Cambridge adminarea = <blank> postalcode=CB4 0WD country= United Kingdom is represented as 296 Cambridge Science Park | Milton Road | | Cambridge | | CB4 0WD | United Kingdom

Table 3–40 Address matching

Address matching rule	Example	Type
Address exact	Flat 1, 296 The Science Park | Milton Road | | Cambridge | Cambridgeshire | CB4 1AB | United Kingdom = Flat 1, 296 The Science Park | Milton Road | | Cambridge | Cambridgeshire | CB4 1AB | United Kingdom	Exact

Table 3–40 (Cont.) Address matching

Address matching rule	Example	Type
Subpremise, premise, postal code starts with, address similar	Flat 1, 296 The Science Park | Milton Road | | Cambridge | Cambridgeshire | CB4 1AB | = Flat 1, 296 The Science Park | Milton Road | | Cambridge | Cambridgeshire | CB4 | United Kingdom	Fuzzy
Premise, no subpremise, postal code starts with, address similar	Flat 1, 296 The Science Park | Milton Road | | Cambridge | Cambridgeshire | CB4 1AB | = 296 The Science Park | Milton Road | | Cambridge | Cambridgeshire | CB4 | United Kingdom	Fuzzy
Subpremise, premise, postal code starts with	Flat 1 | 352 Milton Road | | Cambridge | | CB4 0WD | = 352 Milton Road | Flat 1 | | | | CB4 0WD |	Fuzzy
Premise, no subpremise, postal code starts with	296 Milton Road | Science Park | | Cambridge | | CB4 0WD | = 296 Milton Road | | | | | CB4 0WD |	Fuzzy
Address all words	Flat 1, 296 The Science Park | Milton Road | | Cambridge | Cambridgeshire | CB4 1AB | = | Milton Road | | Cambridge | Cambridgeshire | CB4 1AB | United Kingdom	Fuzzy
Address all words typos	Flat 1, 296 The Science Park | Milton Road | | Cambridge | Cambridgeshire | CB4 1AB | = | Millton Road | | Cambridge | Cambridgeshire | CB4 1AB | United Kingdom	Fuzzy
Address 1 typos, city, country exact or no data	Flat 1, 296 The Science Park | Milton Road | | Cambridge | Cambridgeshire | CB4 1AB | = Science | Mil | | Cambridge | Cambridgeshire | CB4 1AB | United Kingdom	Fuzzy
Address similar, postal code	Flat 1, 296 The Science Park | Milton Road | | Cambridge | Cambridgeshire | CB4 1AB | = Science | Milton | | Cam | Cambridgeshire | CB4 1AB | United Kingdom	Fuzzy
Postal code	Flat 1, 296 The Science Park | Milton Road | | Cambridge | Cambridgeshire | CB4 1AB | = | Arbury Road | | Cambridge | Cambridgeshire | CB4 1AB | United Kingdom	Fuzzy
City and country	Flat 1, 296 The Science Park | Milton Road | | Cambridge | Cambridgeshire | CB4 1AB | = | Arbury Road | | Cambridge | Cambridgeshire | | United Kingdom	Fuzzy
City	Flat 1, 296 The Science Park | Milton Road | | Cambridge | Cambridgeshire | CB4 1AB | = | Arbury Road | | Cambridge | Cambridgeshire | |	Fuzzy
Address similar, first address 1 word	Flat 1, 296 The Science Park | Milton Road | | Cambridge | Cambridgeshire | CB4 1AB | = Datanomic Science Park | Milton Road | Cambridge | Cambridgeshire | | United Kingdom |	Fuzzy
Country	Flat 1, 296 The Science Park | Milton Road | | Cambridge | Cambridgeshire | CB4 1AB | = Datanomic Science Park | Arbury Road | Cambridge | Cambridgeshire | | | United Kingdom	Fuzzy

Table 3–40 (Cont.) Address matching

Address matching rule	Example	Type
Address no data	Flat 1, 296 The Science Park | Milton Road | | Cambridge | Cambridgeshire | CB4 1AB | = | | | | | |	No data
Address conflict	Flat 1, 296 The Science Park | Milton Road | | Cambridge | Cambridgeshire | CB4 1AB | = Datanomic | | | Arbury | | |	Conflict

Table 3–41 Website address

Website address matching rule	Example	Type
Website exact	www.tcnltd.com = www.tcnltd.com	Exact
Website stem exact	www.tcnltd.co.uk = www.tcnltd.com	Fuzzy
Website no data	www.tcnltd.com =	No data
Website conflict	www.abc.com = www.tcnltd.com	Conflict

Phone Number

Phone number matches allow for matches including:

- Exact match
- Last N characters matching

Table 3–42 Phone Number

Phone matching rule	Example	Type
Phone exact	01223456678 = 01223456678	Exact
Phone last N	+44223456678 = 01223456678	Fuzzy
Phone no data	01223456678 =	No data
Phone conflict	01223456678=01684345678	Conflict

Table 3–43 VAT number

VAT number rule	Example	Type
VAT number rule	ABC112345 = ABC112345	Exact
VAT number exact	ABC12345 = ABC112345	Fuzzy
VAT number no data	ABC12345 =	No data
VAT number conflict	ABD2535 = BCD2145	Conflict

Table 3–44 Tax number

Tax number rule	Example	Type
Tax number exact	ABC112345 = ABC112345	Exact
Tax number typos	ABC12345 = ABC112345	Fuzzy
Tax number no data	ABC12345 =	No data
Tax number conflict	ABD2535 = BCD2145	Conflict

The entity matching service outputs fields which give information on the matching of any of the logical identifiers described above, as well as an overall score and an overall rule name. This will allow the consuming application to have more granular information about how the records matched, to use as they wish. Here is an example:

Table 3–45 Comparing Records

Record 1		Record 2	
Name	Widgets and Gadgets Ltd	Name	Gadgets and Widgets Ltd
Subname	Cambridge	Subname	Cambridge
Phone	012234567890	Phone	+4412234567890
Website	www.widgetsandgadgets.com	Website	www.widgetsandgadgets.org
Tax Number	ABC 1234 12	Tax Number	ABC 1234 12
Address1	29 Mill Road	Address1	Flat 3
Address2	Flat 3	Address2	29 Mill Road
City		City	Cambridge
Postal Code		Postal Code	CB1 3GH

Table 3–46 Results of Comparison

Value	Result
ruleattributes	NAME,ADDRESS,PHONE,WEBSITE,TAXNUMBER
matchscore	97
rulename	N090 Full name without suffixes all words out of order, A040 Subpremise, premise, postal code starts with, W020 Website stem exact, P020 Phone last N, T010 Tax number exact, V030
comparisonresults	Name Fuzzy, Address Fuzzy, Website Fuzzy, Phone Fuzzy, Tax Number Exact
nameresult	N090 Full name without suffixes all words out of order
namescore	20
namecategory	Fuzzy
addressresult	A040 Subpremise, premise, postal code starts with
addressscore	50
addresscategory	Fuzzy
phonenumberresult	P020 Phone last N
phonenumberscore	70
phonenumbercategory	Fuzzy
websiteresult	W020 Website stem exact
websitescore	70
websitecategory	Fuzzy
taxnumberresult	T010 Tax number exact
taxnumberscore	100
taxnumbercategory	Exact

Table 3–46 (Cont.) Results of Comparison

Value	Result

*Results that are no data are omitted for brevity

Results that are no data are omitted for brevity.

The comparisonresults output field gives a comma separated list of any logical identifiers that have contributed to the match and the category of the match (i.e. returned a category of Exact or Fuzzy).

The ruleattributes field returns a comma separated list of the logical idenfiers that contributed to the match.

Note: If a field is known never to be populated in the data, then it is possible to "turn off" the compound comparison relating to the logical identifier, so that it does not appear in the rule.

It is also possible to perform matching or elimination of Entity records using custom unique key generation, see Section 3.5, "Using ID Matching."

3.5 Using ID Matching

The ID Matching rules in EDQ-CDS allow matching (or elimination) based solely on custom unique identifiers, without the need for a name match of some kind, irrespective of matching (or not) on other fields. They are performed before, and are completely separate from the rule which matches on the logical identifiers described in the previous sections.

Matching and elimination is provided for Entity and Individual Matching, but not Address Matching.

Note:

- Unique ID (UID) matching is always performed before EID or IEID matching. Therefore, if two records are matched by unique identifiers, they cannot then be eliminated.
- These identifiers are always compared in standardized form; for example, values that differ only in case or additional non-alphanumeric character are considered identical. for example, the following values are identical for the purposes of ID matching:
  - AB123456789
  - ab123-456-789
  - ab12345 6789
  - ab#123456789

3.5.1 Using Unique ID Matching

The UID Match rules are held in the `[I005] UID` and `[E005] UID` match group of the Individual and Entity Match processes respectively. For example, for the match groups for Individual matches are as follows:

- [I005A] Match UID1
- [I005B] Match UID2
- [I005C] Match UID3

To use these rules, map the required data in the records to one or more of the **uid** attributes. The matching rules will always match two records sharing a common unique identifier, even if none of the other attributes match.

Note:

- The uid attributes accept multiple values in the form of a pipe delimited list. A match will be returned between two records if any one of a multiple set of attribute values is matched.
- Matching between uid attributes is not possible, for example, uid1 values cannot be matched with uid2 or uid3 values.

Example

The Passport Number field in a series of records is configured as the uid1 attribute. Therefore, the following records are returned as a match:

Record ID	First Name	Last Name	uid1 (Passport Number)	Match?
1	Fred	Smith	12345678	Yes
2	John	Doe	12345678	Yes

The following records with multiple values in the uid1 field are also matched:

Record ID	First Name	Last Name	uid1 (Passport Number)	Match?
1	Fred	Smith	12312312 | 67867867	Yes
2	John	Doe	67867867 | 23423423	Yes

The SSN field for the same set of records is configured as the uid2 attribute. The uid1 and uid2 fields are not cross matched; even though the uid1 value of Record 1 matches the uid2 value of Record 2:

Record ID	First Name	Last Name	uid1 (Passport Number)	uid2 (SSN)	Match?
1	Fred	Smith	12312312	67867867	No
2	John	Doe	67867867	12312312	No

3.5.2 Using Elimination ID Matching

The Elimination ID (EID) Match rules are held in the [ELIM015] EID ELIMINATIONS group of the Entity and Individual Match processes:

- [ELIM015A] ELIMINATE EID1
- [ELIM015B] ELIMINATE EID2
- [ELIM015C] ELIMINATE EID3

To use these rules, map the required data in the records to one or more of the eid attributes. The EID matching rules will always return a "No Match" result for two records that do not share a common value in an eid attribute, even if all other attributes match. The exception to this is if the two records are matched using a uid attribute, as UID matching is performed before EID matching.

Note:

- eid attributes accept multiple values in the form of a pipe delimited list. A "No Match" result will be returned between two records if none the values in an attribute are matched.
- Eliminating possible matches by comparing values between different eid attributes is not possible, for example, eid1 values cannot be compared with eid2 or eid3 values.

Example

The SSN field in a series of records is configured as the eid1 attribute. Therefore, the following records are eliminated as a possible match:

Record ID	First Name	Last Name	eid1 (SSN)	Eliminate?
1	John	Doe	12345678	Yes
2	John	Doe	87654321	Yes

The following records with multiple values in the eid1 field are also eliminated as a possible match, as none of the values match:

Record ID	First Name	Last Name	eid1 (SSN)	Eliminate?
1	John	Doe	12312312 | 23423423	Yes
2	John	Doe	45645645 | 67867867	Yes

The Passport field for the same set of records is configured as the eid2 attribute. The eid1 and eid2 fields are not compared, and therefore a "No Match" result is returned and the records are eliminated as a possible match:

Record ID	First Name	Last Name	eid1 (SSN)	eid2 (Passport Number)	Eliminate?
1	John	Doe	12312312	67867867	Yes
2	John	Doe	67867867	12312312	Yes

Finally, there are two identical values in the eid1 fields of the following records, and therefore they are *not* eliminated as a possible match:

Record ID	First Name	Last Name	eid1 (SSN)	Eliminate?
1	John	Doe	12312312 | 23423423	No
2	John	Doe	45645645 | 12312312	No

3.5.3 Using Inverted Elimination ID Matching

The Inverted Elimination ID (IEID) Match rules are held in the `INVERTED EID ELIMINATIONS` group of the Entity and Individual Match processes:

Inverted ID matching provides similar functionality to Elimination Ids (EIDs) but produces a "No match" result when the identifier values are the same. Inverted ID matching allows you to eliminate matches where records share a common value.

To use these rules, map the required data in the records to one or more of the `ieid` attributes. The IEID matching rules will always return a "No Match" result for records where the inverted EID (IEID) values are the same.

3.6 Using Matching with Customer-Added Attributes

Matching with customer-added string and date attributes improves how you can configure EDQ and reduces the need to customize the EDQ-CDS configuration for attributes not present on the standard interface.

The Individual Candidates and Entity Candidates interfaces each contain six custom string and three custom date attributes. The Matches interface contains custom result, category, and score attributes for each custom string and custom date.

3.6.1 Standardization

Custom strings can be specified as type `identifier` or `text`, which affects how they are standardized: `identifier` custom strings are stripped of non-alphanumeric characters and converted to upper case, while `text` custom strings are just normalized.

This behavior is specified in the run profile as follows:

```
phase.*.process.*.customstringNtype = text
```

and can be overridden in real-time on a per-message basis as follows:

```
<dn:request customstringNtype="identifier">
```

Custom dates are standardized the same way, as a conversion to the `date` data type.

3.6.2 Matching

Custom attributes can optionally be used during matching (by default no matching is performed on custom attributes) irrespective of whether or not they have been used for keying (see Keys for Custom Attributes).

There are two ways custom attributes can be matched:

- Exact only
- Exact and fuzzy

There are two compound comparisons for each custom attribute:

- `customstringNexact/customdateNexact`
- `customstringNfuzzy/customdateNfuzzy`

Therefore the enablement and type of matching performed for each custom attribute, and the corresponding weighting, is specified in the run profile by using the relevant 'exact' or 'fuzzy' parameters for each of these compound comparisons, for example:

```
phase.*.process.Match\ -\ Individual.overallscore.customstring1exact.enabled = Y
phase.*.process.Match\ -\ Individual.overallscore.customstring1exact.weighting = 1
phase.Individual\ Match.process.*.overallscore.customstring1fuzzy.enabled   = N
```

```
phase.Individual\ Match.process.*.overallscore.customstring1fuzzy.weighting = 1
```

That is, in order to match on any given custom attribute, either the corresponding 'exact' or 'fuzzy' compound comparison should be enabled, but not both.

These settings can also be overridden in real-time on a per-message basis as follows:

```
<dn:request
   overallscore.customstring1exact.enabled="Y"
   overallscore.customstring1exact.weighting="1"
   overallscore.customstring1fuzzy.enabled="N"
   overallscore.customstring1fuzzy.weighting="1"
>
```

3.7 Using Address Matching

The rules for matching addresses include the use of pre-matching transformations and various matching comparisons in order to handle variance between different representations of what may be the same address, for example:

- Addresses containing abbreviated terms or suffixes.
- Character order and spelling differences/errors in addresses.
- Addresses with different levels of completeness.
- Addresses where extracted premise and sub-premise match, and other components of the address are in a different order or missing on one side.

The following table lists all of the rules provided:

Address Match Rule Code	Address Match Rule Description
[A010]	Address exact, postal code exact
[A020]	Address exact, no postal code
[A030]	Address lines 1 and 2 exact, city exact, postal code exact
[A040]	Address lines 1 and 2 exact, city exact, postal code starts with
[A050]	Address all words, subpremise exact, premise exact, postal code exact
[A060]	Address all words, subpremise exact, premise exact, postal code no conflict
[A070]	Address 1 exact, address 2 no conflict, subpremise exact, premise exact postal code exact
[A080]	Address 1 exact, address 2 no conflict, subpremise exact, premise exact, postal code starts with
[A090]	Address 1 exact, address 2 no conflict, subpremise exact, premise exact, postal code no conflict
[A100]	Address all words typos, subpremise exact, premise exact, postal code exact
[A110]	Address all words typos, subpremise exact, premise exact, postal code no conflict
[A120]	Address 1 exact, address 2 no conflict, postal code exact
[A130]	Address 1 exact, address 2 no conflict, postal code starts with
[A140]	Address 1 exact, subpremise exact, premise exact, postal code exact
[A150]	Address 1 exact, subpremise exact, premise exact, postal code starts with

Address Match Rule Code	Address Match Rule Description
[A160]	Address 1 exact, subpremise no conflict, premise no conflict, postal code exact
[A170]	Address 1 exact, subpremise no conflict, premise no conflict, postal code starts with
[A180]	Address all words, subpremise no conflict, premise no conflict, postal code exact
[A190]	Address all words, subpremise no conflict, premise no conflict, postal code no conflict
[A200]	Address 1 all words, subpremise exact, premise exact, postal code exact
[A210]	Address 1 all words, subpremise exact, premise exact, postal code starts with
[A220]	Address 1 all words, subpremise no conflict, premise no conflict, postal code exact
[A230]	Address 1 all words, subpremise no conflict, premise no conflict, postal code starts with
[A240]	Address1 common string 7+, subpremise exact, premise exact, postal code exact
[A250]	Address all words, postal code exact
[A260]	Address similar, subpremise exact, premise exact, postal code exact
[A270]	Address 1 all words, address 2 no conflict, postal code exact
[A280]	Address 1 all words, address 2 no conflict, postal code starts with
[A290]	Address all words typos, postal code exact
[A300]	Address 1 exact, subpremise exact, premise exact, postal code no conflict
[A310]	Address 1 all words, subpremise exact, premise exact, postal code no conflict
[A320]	Address 1 exact, postal code exact
[A330]	Address 1 exact, postal code starts with
[A340]	Subpremise exact, premise exact; postal code exact
[A350]	Subpremise exact, premise exact, postal code starts with
[A360]	Address all words
[A370]	Address all words typos
[A380]	Address similar; postal code
[A390]	Address similar; first address one word

The following table provides examples of matches by Match Rule Code only, with the key fields highlighted in bold text where required:

Address Match Rule Code	Address Component	Record	Matched Record
[A010]	address1	901 GOLF CLUB RD	901 GOLF CLUB RD
	city	WESTWOOD	WESTWOOD
	subadminarea	PLUMAS	PLUMAS
	adminarea	CA	CA
	postalcode	96137	96137

Address Match Rule Code	Address Component	Record	Matched Record
	country	US	US
[A020]	As for [A010], but the **postalcode** field in both records is blank.		
[A030]	address1	1201 BEECH ST	1201 BEECH ST
	address2	APT 104F	APT 104F
	city	PALO ALTO	PALO ALTO
	subadminarea	**SANTA CLARA**	**SAN MATEO**
	adminarea	CA	CA
	postalcode	94303	94303
	country	US	US
[A040]	As [A030], except the v field in one address starts with the same characters as the other, but is not identical.		
[A050]	address1	**5 Hogskoleringen**	**Hogskoleringen 5**
	city	Trondheim	Trondheim
	adminarea		SØR-TRØNDELAG
	postalcode	**7491**	**7491**
	country	Norway	Norway
[A060]	As [A050], except one or both of the **postalcode** fields are blank.		
[A070]	address1	**Heinrichboeckingstr 10-14**	**Heinrichboeckingstr 10-14**
	address2	Service Zentrum Merzig	
	city	Saarbrücken	Saarbrücken
	adminarea		SAARLAND
	postalcode	66121	66121
	country	Germany	Germany
[A080]	Same as [A070], except the postalcode field in one address starts with the same characters as the **postalcode** field in the other, but is not identical.		
[A090]	Same as [A070], except one or both of the `postalcode` fields are blank.		
[A100]	address1	**HOGSKOLERINGE 5**	**HOGSKOLERINGEN 5**
	city	Trondheim	Trondheim
	postalcode	9491	9491
	country	Norway	Norway
[A110]	Same as [A100], except one or both of the **postalcode** fields are blank.		
[A120]	address1	**Marshfield Bank**	**Marshfield Bank**
	address2	**WOOLSTANWOOD**	
	city	Crewe	Crewe
	postalcode	**CW28UY**	**CW28UY**
	country	UK	UK

Address Match Rule Code	Address Component	Record	Matched Record
[A130]	Same as [A120], except the **postalcode** field in one address starts with the same characters as the **postalcode** field in the other, but is not identical.		
[A140]	address1	**Apt Y302**	**APT Y302**
	address2	**1605** Sherringtowne Ave	**1605** Sherington Ave
	city	NEWPORT BEACH	NEWPORT BEACH
	adminarea	Orange	Orange
	postalcode	**92663-9087**	**92663-9087**
	country	US	US
[A150]	Same as [A140], except the **postalcode** field in one address starts with the same characters as the **postalcode** field in the other, but is not identical.		
[A160]	address1	1728 **Corporate Xing**	1728 **Corporate Xing**
	address2	**Suite1**	
	city	O Fallon	O Fallon
	adminarea	ILLINOIS	IL
	postalcode	62269-3734	62269-3734
	city	US	US
[A170]	Same as [A160], except the **postalcode** field in one address starts with the same characters as the **postalcode** field in the other, but is not identical.		
[A180]	address1	Block 16	16 Dunsinane Ave
	address2	Dunsinane Avenue	
	address3	Dunsinane Industrial Estate	
	city	Dunsinane	Dunsinane
	postalcode	DD23QT	DD23QT
	country	UK	UK
[A190]	As [A180], except one or both of the **postalcode** fields are blank.		
[A200]	address1	**26701 QUAIL CRK**	**26701 QUAIL CRK APT 107**
	address2	**APT 107**	
	city	ALISO VIEJO	LAGUNA HILLS
	postalcode	92656-1089	92656-1089
	country	US	US
[A210]	Same as [A200], except the **postalcode** field in one address starts with the same characters as the **postalcode** field in the other, but is not identical.		
[A220]	address1	**Folkes Road**	**Unit 12 Folkes Road**
	address2	Hayes Trading Estate	Lye
	address3	Lye	
	city	Stourbridge	Stourbridge
	postalcode	**DY98RN**	**DY98RN**
	country	UK	UK

Address Match Rule Code	Address Component	Record	Matched Record
[A230]	Same as [A220], except the **postalcode** field in one address starts with the same characters as the **postalcode** field in the other, but is not identical.		
[A240]	address1	**101/61 NAWANAKORN INDUSTRY**	**101/61 NAVANAKORN INDUSTRY**
	address2	SELFLEMENT PHAHONYOTHIN	PAHOLYOTHIN KLONGNUENG
	city	KLONGLAUNG	KHLONG LUANG
	postalcode	**12120**	**12120**
	country	Thailand	Thailand
[A250]	address1	Blyth House	Blyth House
	address2	**130** Hordern Road	Hordern Road
	city	Wolverhampton	Wolverhampton
	postalcode	WV60HS	WV60HS
	country	UK	UK
[A260]	address1	**21001** State Route 739	**21001** Sr Rt **739**
	address2	7	
	city	Raymond	Raymond
	postalcode	**43067**	**43067**
	country	United States	United States
[A270]	address1	Lancaster House Aviation Way	Aviation Way
	address2		Southend Airport
	city	SOUTHEND ON SEA	SOUTHEND ON SEA
	postalcode	SS26UN	SS26UN
	country	UK	UK
[A280]	Same as [A270], except the **postalcode** field in one address starts with the same characters as the **postalcode** field in the other, but is not identical.		
[A290]	address1	**Blythe House**	**Blyth House**
	address2	130 **Hordern Road**	**Hordern Road**
	city	Wolverhampton	Wolverhampton
	postalcode	**WV60HS**	**WV60HS**
	country	UK	UK
[A300]	Same as [A140], except one or both of the **postalcode** fields are blank.		
[A310]	Same as [A200], except one of both of the **postalcode** fields are blank.		
[A320]	address1	**Network House**	**Network House**
	address2	1 Ariel Way	Wood Lane
	city	London	London
	postalcode	**W127SL**	**W127SL**
	country	UK	UK

Address Match Rule Code	Address Component	Record	Matched Record
[A330]	Same as [A320], except the **postalcode** field in one address starts with the same characters as the **postalcode** field in the other, but is not identical.		
[A340]	address1	College Business Park	College Business Park
	address2	Park	Coldhams Lane
	city	Cambridge	
	postalcode	**CB13HD**	**CB13HD**
	country	United Kingdom	United Kingdom
[A350]	Same as [A340], except the **postalcode** field in one address starts with the same characters as the **postalcode** field in the other, but is not identical.		
[A360]	address1	938 Miller St	Medical Ctr Blvd
	address2	**Medical Center Boulevard**	
	city	Winston Salem	Winston- Salem
	postalcode	27157	27157
	country	United States	United States
[A370]	address1	**Humberstone Avenue**	24 **Humberston Avenue**
	address2	**Humberstone**	**Humberston**
	city	GRIMSBY	GRIMSBY
	postalcode	DN364SX	DN364SP
	country	UK	UK
[A380]	address1	5**Sidings Court**	Greyfriars House
	address2	White Rose Way	**Sidings Court**
	city	DONCASTER	DONCASTER
	postalcode	**DN45NU**	**DN45NU**
	country	UK	UK
[A390]	address1	120 **Howard** St	120 **Howard** St
	address2		STE 200
	city	San Fransisco	San Fransisco
	adminarea	CA	CA
	postalcode	94105-1622	94105-1615
	country	United States	United States

Note: Unlike Individual and Entity matching, Address Matching does not make use of the compound comparison match functionality, since it does not lend itself to splitting the matching between separate logical identifiers for matching in the same way.

4

Installing and Using Data Quality Health Check

This chapter describes how you can use the EDQ-CDS Data Quality Health Check functionality.

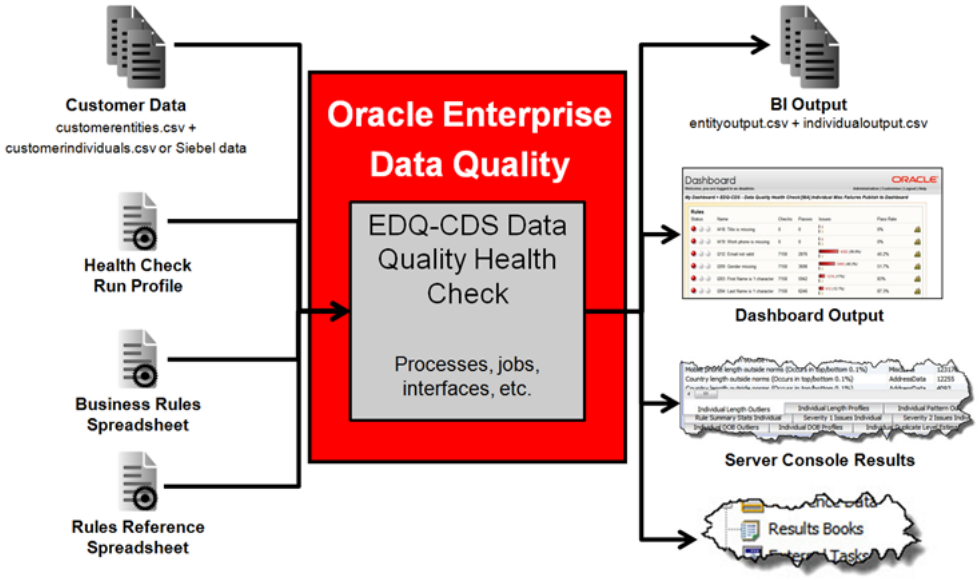
This chapter includes the following sections:

Data Quality Health Check extends the capability of the EDQ-CDS, allowing you to perform batch data quality checking of your data before it has been normalized or standardized. The results can then be viewed in the Server Console, a Business Intelligence (BI) tool, in the EDQ Results books, or published to the Dashboard as required. As a component of EDQ-CDS, Data Quality Health Check can be integrated with Siebel or used in stand-alone mode.

EDQ-CDS Data Quality Health Check will primarily be of use to anyone requiring a view of the quality of raw data, from Data Stewards who require a data-level view of data quality issues, to Operations Analysts and Executives who require Dashboard information for analysis, reporting and planning purposes. Additionally, it is useful for Data Professionals that want to analyze the technical aspects of data, and to EDQ-CDS users seeking to ensure their CDS processes are performing efficient deduplication.

4.1 Architecture

The following illustrates how you can use EDQ-CDS Data Quality Health Check to process your data and view the results:



4.1.1 Multiple Child Entities

Some data will feature multiple child entities, for example, more than one address might be assigned to each record. When such records are processed and passed to EDQ, one record per child is created.

Therefore, the Data Quality Health Check results often list a greater number of records than are initially taken in. It is important to remember this when viewing results in Server Console or Dashboard.

4.2 Installing Data Quality Health Check

The section explains how to install Data Quality Health Check. While Data Quality Health Check is part of the EDQ-CDS distribution, it is not necessary to fully configure EDQ-CDS in order to use Data Quality Health Check. EDQ Health Check has the same prerequisites as the EDQ-CDS.

Siebel integrations require the installation of the Siebel Connector Release 12*c* (12.2.1). For more information, see *Oracle Fusion Middleware Installing and Customizing Enterprise Data Quality Customer Data Services Pack*.

4.2.1 Installation Components

The components necessary to install Data Quality Health Check are pre-loaded in EDQ and are therefore automatically installed with EDQ.

The EDQ-CDS Health Check components are:

- `edq-cds-data-quality-health-check.dxi` - the packaged EDQ project containing the EDQ-CDS data quality services.
- `dq-health-check-business-rules-individual.xls` - Individual Business Rules spreadsheet, which defines the data quality checks performed for individuals.
- `dq-health-check-business-rules-entity.xls` - Entity Business Rules spreadsheet, which defines the data quality checks performed for entities.
- `edq-cds-data-quality-health-check.properties` - the default Run Profile.

- customerentities.csv - Sample Entity data.
- customerindividuals.csv - Sample Individual data
- rulesreference.xls - Spreadsheet categorizing the error codes present in the Business Rules spreadsheets.

4.2.2 Installing the Software

If you have installed EDQ-CDS, then Data Quality Health Check is installed and no further installation tasks are necessary.

To install Data Quality Health Check without the presence of EDQ-CDS, use the following procedure:

1. Start the **EDQ Director client**, and log on as a user with the permission to create projects (Administrator or Project Owner).
2. Right-click on the server name and select **Open Server Package File**. Open the CDS folder and select the edq-cds-data-quality-health-check.dxi file.
3. Drag the whole **EDQ-CDS - Data Quality Health Check** project onto the **Projects** node.
4. Right-click on the .dxi file, and select **Close Package File**.

4.2.3 Verifying the Installation

Data Quality Health Check comes with two sample .csv files in the landingarea/dqhealthcheck folder. These files can be used to test the installation is working correctly.

The sample files are:

- customerentities.csv - Sample Entity data.
- customerindividuals.csv - Sample Individual data.

The default jobs provided with Data Quality Health Check are configured to run against these files.

To verify the installation, run either (or both) of the **CSV Batch Entity Data Quality Health Check** or **CSV Batch Individual Data Quality Health Check** jobs in Server Console, remembering to select the edq-cds-data-quality-health-check.properties Run Profile.

Note:

Data Quality Health Check uses its own internal reference data, and therefore does not need the CDS Initialize project to be run before it is used.

Do not attempt to run any of the Siebel jobs manually; these jobs are designed to be invoked automatically by the Siebel Connector.

Check the Event Log and Results in Server Console to ascertain whether the job (or jobs) have completed correctly. If so, then the installation has been successful.

Finally, purge the results of the job or jobs in the Server Console and Dashboard:

- **Server Console**:

Select the **Results** view, right click the job in the **Job History** area, select the **Purge data for run label [Name of Run Label]** option.

- **Dashboard**:

  Open Dashboard Administration, expand the **Audit** tree in the **Audits & Indexes** area, right click on the **Data Quality Health Check** audit and select **Purge**.

4.3 Configuring Data Quality Health Check

This section explains how to configure Data Quality Health Check.

4.3.1 Configuring Business Rules

The Business Rules are set in two `.xls` files supplied with Data Quality Health Check, located in the `oedq.home/business rules` folder. To edit these `.xls` files, you must move them to `oedq_local_home/business rules`.

- `dq-health-check-business-rules-individual.xls` - Individual Business Rules spreadsheet.
- `dq-health-check-business-rules-entity.xls` - Entity Business Rules spreadsheet.

There is an additional spreadsheet - `rulesreference.xls` - in the `oedq_local_home/landingarea/dqhealthcheck` folder which has two main functions: it is used to control which rules in the Business Rules spreadsheets are used when running Data Quality processes, and also to construct rules statistics.

Note : By default, the Individual and Entity rules that are used by EDQ-AV are enabled in the `rulesreference.xls` spreadsheet. If EDQ-AV is not installed these rules must be disabled to prevent inaccurate reporting in the Dashboard.

The **Enabled** column in the `rulesreference.xls` spreadsheet controls which rules are enabled and which are disabled, the two possible values being `yes` and `no`. Therefore, if any existing rules are edited or new rules added to the Business Rules spreadsheets, the changes must be reflected in the `rulesreference.xls` sheet. Any changes made must preserve the separation of rule types, which object (Individual or Entity) they relate to, and their associated rule and error codes.

The rules fall into the following categories:

- Population checks – Check that a field is not blank. For example, `ER205 - Check if Name is missing`.
- List checks - Check that the data contains only values from a specified list. For example, `IR202 - Check if Upper Case gender is a valid value`.
- Length checks - Check that the data is of a specified length,. For example, `IR203 - Check first name is > 1 char`.
- Format checks - Check that the data conforms to a pattern or regular expression. For example, `IR212 - Check if email is valid format`.
- Contains checks - Check that the data contains a value from a list; for example `IR428 - Check if full name is clear of entity hints`.
- Suspect data checks - Check that the data exhibits any common data entry "cheats". For example, `ER411 - Check if unusual characters in name`.

- Value checks – Check that the field value is in the correct range. For example, `IR430 - Check if DOB is very old (<1900).`
- Dependent attribute checks – Check that two attribute values are consistent, for example, if the value in one attribute is dependent on the value in another attribute. For example, `IR302 - Check if gender and title are consistent.`
- Duplicate checks - Compare combinations of data attributes to estimate potential levels of record duplication. This is not full EDQ-CDS matching, and therefore is designed to run in a fraction of the time. Examples of comparisons include:
  - `IR401 - Check if fname address1 are flagged dupe`
  - `IR403 - Check if fname email are flagged dupe`
  - `IR408 - Check if lname tax no are flagged dupe`

For information on customizing existing and creating new Business Rules using these spreadsheets, see the "Defining Business Rules" topic in the *Oracle Enterprise Data Quality Director Online Help*.

4.3.2 Configuring the Run Profile

The `edq-cds-data-quality-health-check.properties` Run Profile is divided into the following sections:

4.3.2.1 Publish to Dashboard Setting

This setting controls whether the results of the Health Check jobs are published to the Dashboard:

```
phase.Publish\ to\ Dashboard.enabled = yes
```

The default value is `yes`. Change to `no` to prevent the results being published.

> **Note :** The value must always be in lower case, `yes` or `no`.

4.3.2.2 Input Source Location, Separator and Encoding Settings

These settings specify the source of the input files for individual and entity data, the field separator used, and the encoding employed. The default settings are included as in the following:

```
phase.*.snapshot.*.Entity_Input_CSV_File_Location =
\\dqhealthcheck\\customerentities.csv
phase.*.snapshot.*.Entity_Input_CSV_File_Field_Separator = \,
phase.*.snapshot.*.Entity_Input_CSV_File_Encoding = UTF-8
phase.*.snapshot.*.Individual_Input_CSV_File_Location =
\\dqhealthcheck\\customerindividuals.csv
phase.*.snapshot.*.Individual_Input_CSV_File_Field_Separator = \,
phase.*.snapshot.*.Individual_Input_CSV_File_Encoding = UTF-8
```

The file and folder location specified *must* be in the `landingarea` folder.

The encoding of the input file must be a valid encoding for EDQ delimited text Data Stores. The escape character - backslash "\" - must be used if the desired separator is a reserved character, for example, a comma. A list of valid encoding formats can be found in the **Edit Data Store** dialog in EDQ.

4.3.2.3 Publish Results as CSV Setting

This setting controls whether the results of the Health Check jobs are published in the form of a `.csv` file for use in a BI tool:

```
phase.Export\ BI\ Data.enabled = no
```

The default value is `no`. Set to `yes` to publish the data to the `.csv` file.

> **Note :** The value must always be in lower case, `yes` or `no`.

4.3.2.4 Export File Location, Separator and Encoding Settings

If export is enabled, these settings specify the destination of the exported file, the field separator and encoding. The default settings are included as in the following:

```
phase.*.Export.*.Entity_Output_CSV_File_Location =
\\dqhealthcheck\\entityoutput.csv
phase.*.Export.*.Entity_Output_CSV_File_Field_Separator = \,
phase.*.Export.*.Entity_Output_CSV_File_Encoding = UTF-8
phase.*.Export.*.Individual_Output_CSV_File_Location =
\\dqhealthcheck\\individualoutput.csv
phase.*.Export.*.Individual_Output_CSV_File_Field_Separator = \,
phase.*.Export.*.Individual_Output_CSV_File_Encoding = UTF-8
```

> **Note :** The encoding of the export file must be valid for EDQ delimited text Data Stores. A list of valid encoding formats can be found in the **Edit Data Store** dialog in EDQ.

4.3.2.5 Default Country Code for AV

If EDQ-AV is installed, this setting should be assigned the ISO two-character country code to be used by default. For example, if the country code is not specified in the data supplied:

```
phase.*.process.*.Default\ AV\ Country\ Code
```

The default value is `US`. Any codes that are entered here are expected to comply with the ISO-3166-1-alpha-2 specification.

4.3.2.6 Results Book Settings

To create EDQ Results Books populated with Individual and/or Entity profiling data, uncomment the following settings.

> **Note :** The first six lines are for the Individual Profiling Results book, and the last two are for the Individual Rules Results book. It is possible to populate one or both of these books as required.

For Individual data, these settings will populate the Individual Profiling Results Book with drillable results of all profilers and the Individual Rules Results Book with a drillable view of rule failures.

```
phase.Profile\ Individual\ Misc\ Data.enabled = no
phase.Profile\ Individual\ Misc\ Data\ With\ Results\ Book.enabled = yes
phase.Profile\ Individual\ Address\ Data.enabled = no
phase.Profile\ Individual\ Address\ Data\ With\ Results\ Book.enabled = yes
phase.Profile\ Individual\ Alt\ Phone Data.enabled = no
```

```
phase.Profile\ Individual\ Alt\ Phone\ Data\ With\ Results\ Book.enabled = yes
phase.Process\ Rule\ Failures\ to\ Outputs.enabled = no
phase.Process\ Rule\ Failures\ to\ Outputs\ With\ Results\ Book.enabled = yes
```

For Entity data, these settings will populate the Entity Profiling Results Book with drillable results of all profilers and the Entity Rules Results Book with a drillable view of the rule failures:

```
phase.Profile\ Entity\ Misc\ Data.enabled = no
phase.Profile\ Entity\ Misc\ Data\ With\ Results\ Book.enabled = yes
phase.Profile\ Entity\ Address\ Data.enabled = no
phase.Profile\ Entity\ Address\ Data\ With\ Results\ Book.enabled = yes
phase.Profile\ Entity\ Alt\ Phone\ Data.enabled = no
phase.Profile\ Entity\ Alt\ Phone\ Data\ With\ Results\ Book.enabled = yes
phase.Make\ Analysis\ and\ Server\ Console\ Output.enabled = no
phase.Make\ Analysis\ and\ Server\ Console\ Output\ With\ Results\ Book.enabled =
yes
```

4.3.2.7 Staged Data Visibility Settings Within Server Console

These settings control which Staged Data items are visible in Server Console.

The first setting - `stageddata.*.visible = no` - makes all Staged Data items invisible by default. The remaining settings then make specific Staged Data items visible.

By default, detailed data in the DQ Health Check Analysis Output tab in the Server Console Results screen is hidden. This is because the level of detail is seldom required for most purposes. To view this data, set the following properties in the Run Profile to yes:

- `stageddata.Individual\ DQ\ Health\ Check\ Analysis\ Output.visible =`
- `stageddata.Entity\ DQ\ Health\ Check\ Analysis\ Output.visible =`

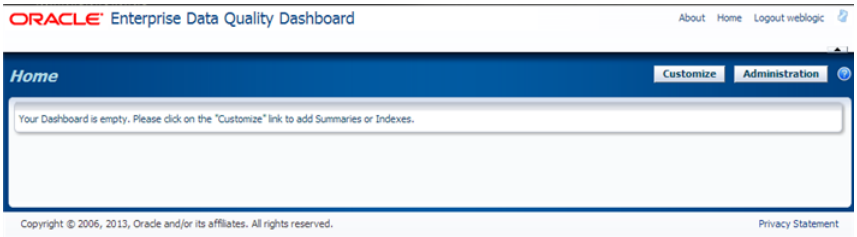
4.4 Configuring the Dashboard

By default, the Health Check results are published to the Dashboard.

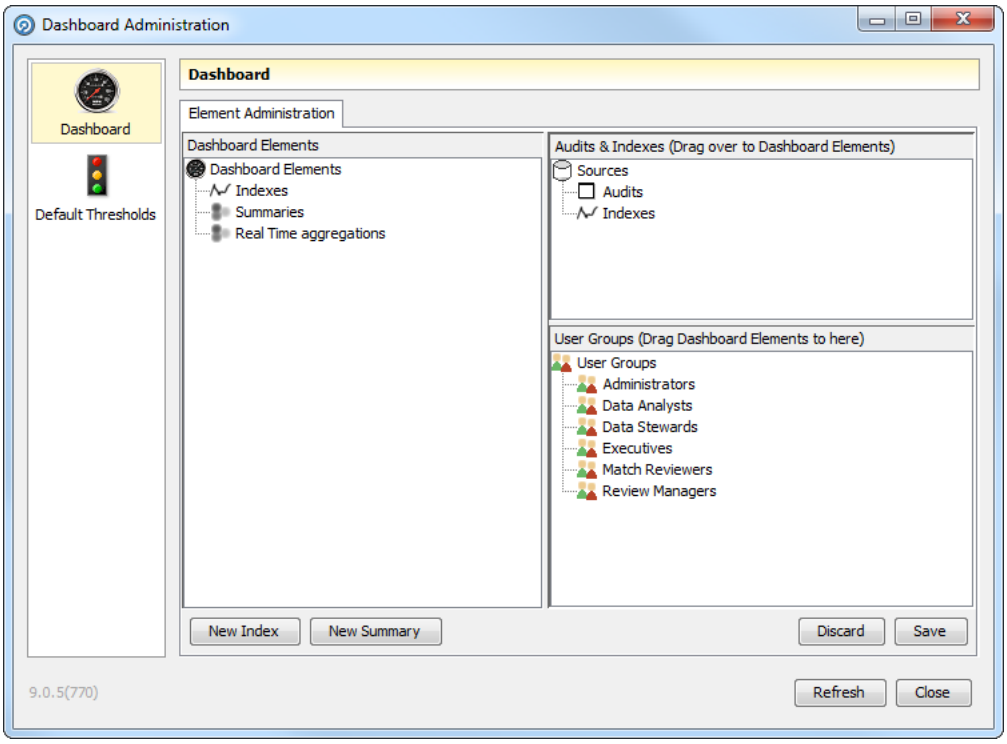
The Dashboard is accessed from the EDQ Launchpad:

To configure Health Check results on the Dashboard, use the following procedure:

1. Open the Dashboard.
2. On the main Dashboard, click **Administration**.



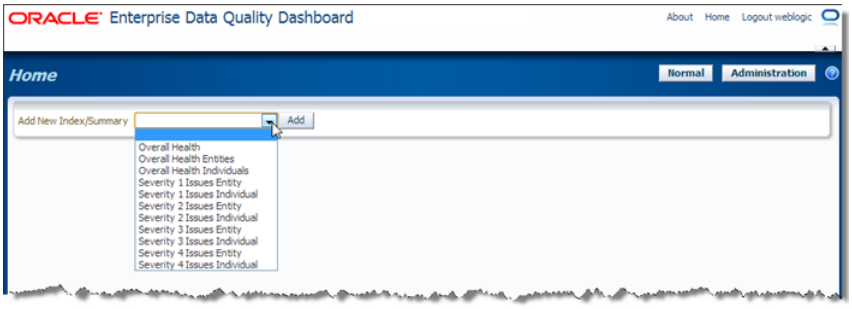
The Dashboard Administration is displayed:



3. Create the Summaries and Indexes as required.

> **Note :** Any rules added to the Summaries should correspond with those enabled in the `rulesreference.xls` spreadsheet. If a disabled rule is included in a Summary or Index it will always be red-flagged, regardless of the results of enabled rules.

4. Return to the Dashboard and click **Customize**.
5. Select the Data Quality results to view in the **Add New** drop-down field. For example:



6. Click **Add**. The selected item is added to the Home view.

Once this configuration procedure is complete, it is possible to choose which Summaries and Indexes to add to the Initial view, to drill down into the results. For full details of how to do this, see *Oracle Enterprise Data Quality Dashboard Online Help*.

4.4.1 Example: Dashboard By Severity

This is an example of a Dashboard configuration that groups rules into Summaries by severity, and then into Indexes.

The first letter of the Health Check rule audit codes indicates the record type ("I" for Individual and "E" for Entity), and the first number indicates the severity level (1, 2, 3 or 4). For example, code E203 is an Entity rule with a severity level of 2.

Create eight summaries to contain the Individual and Entity rule results for severity levels 1 to 4:

- Severity 1 Issues Individual
- Severity 2 Issues Individual
- Severity 3 Issues Individual
- Severity 4 Issues Individual
- Severity 1 Issues Entity
- Severity 2 Issues Entity
- Severity 3 Issues Entity
- Severity 4 Issues Entity

Then create the following Indexes:

Name	Contents
Overall Health Individuals	Contains all the Individual Summaries.
Overall Health Entities	Contains all the Entity Summaries.
Overall Health	Contains the Individual and Entity Summaries.

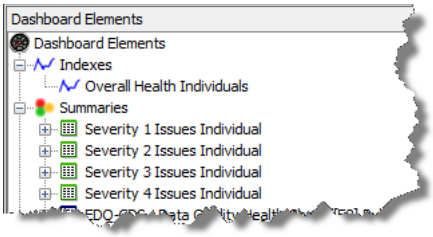
4.4.1.1 Creating the Summaries

1. Open EDQ Dashboard, and click **Administration** to open the **Dashboard Administration** window.
2. Click **New Summary**.
3. Enter **Severity 1 Issues Individual** in the **Add Summary** pop-up.
4. Click **OK**. The new Summary is displayed in the **Summaries** node of the **Dashboard Elements** area.
5. In the **Audits and Indexes** area, expand the **Audits** branch, then expand the **EDQ-CDS – Data Quality Health Check/[I8A] Individual Misc Failures Publish to Dashboard** branch.
6. Click and drag **I101** and **I102** from the **[I8A] Individual Misc Failures Publish to Dashboard** audits list to the **Severity 1 Issues Individual** Summary.
7. Click and drag the **Severity 1 Issues Individual** Summary to the **Administrators** node in the **User Group** area.
8. Click **Save**.
9. Repeat for the remaining summaries.

4.4.1.2 Creating the Indexes

This example assumes all the Summaries detailed in the previous sections have been configured.

1. Open EDQ Dashboard, and click **Administration** to open the **Dashboard Administration** window.
2. Click **New Index**.
3. Name the new Index "Overall Health Individuals".



4. Click and drag the following Summaries into the new Index:
   - Severity 1 Issues Individual
   - Severity 2 Issues Individual
   - Severity 3 Issues Individual
   - Severity 4 Issues Individual
5. Click **Save**.
6. Repeat for the remaining Indexes.

4.4.2 Example - Dashboard By Business Function

This is an example of a Dashboard configuration that groups rules into Summaries by Business Function.

1. Create the following Summaries:

Name	Contents
Account	▪ Name Details ▪ Identifiers ▪ Identifier outliers ▪ Address details ▪ Address detail outliers ▪ Potential duplicates
Contact	▪ Name details ▪ Identifiers ▪ Identifier outliers ▪ Address details ▪ Address detail outliers ▪ Potential duplicates

The rules to be included in each Summary are detailed in Section 4.8, "Dashboard Example Summaries." Ensure that all these rules are enabled.

2. Create the following Indexes:

Name	Contents
Overall Health Account	Containing all the Account-based Summaries.
Overall Health Contacts	Containing all the Contact-based Summaries.
Overall System Health	Containing all the Summaries you created.

4.5 Running Health Check Jobs and Viewing Results

This section describes how to run Health Check jobs and view the results.

4.5.1 Running a Health Check

Health Check jobs can be run either from Siebel, in stand-alone mode from Server Console, or in EDQ-CDS.

If running from Server Console, it may be necessary to prepare the data first.

There are six Health Check jobs:

- Perform Entity Technical Analysis
- Perform Individual Technical Analysis
- CSV Batch Entity Data Quality Health Check
- CSV Batch Individual Data Quality Health Check
- Database Batch Entity Health Check
- Database Batch Individual Health Check

4.5.1.1 Using the Siebel-Attached Mode

Before Health Check can be used with Siebel, the Siebel Connector must be installed and Siebel must be configured accordingly. For more information, see *Oracle Fusion Middleware Installing and Customizing Enterprise Data Quality Customer Data Services Pack*.

To run a Health Check job in Siebel, open Server Manager and access the Data Quality Manager component. The two jobs that should be run from Siebel are:

- Siebel Batch Account Health Check
- Siebel Batch Contact Health Check

> **Note :** The other Health Check jobs should not be configured to run from Siebel. It is possible to do this, but they will not return any results. They must always be run from Server Console or EDQ.

Additionally, any settings changed in the Run Profile must also be changed in the `dnd.properties` file to ensure that the changes are accurately reflected in a Siebel batch run.

4.5.1.2 Using the Stand-Alone Mode

The Technical Analysis and Run Entity/Individual Quality Health Check jobs are designed to be run from EDQ or Server Console.

If the data to be checked can be provided in exactly the same format as the sample data files (for example, `.csv` files with column headings as described in Section 4.7, "Understanding Data Interfaces"), simply save these files to the `landingarea\dqhealthcheck` folder using the same file names as (overwriting) the sample data files.

However, if the data is provided in a different format EDQ should be configured to use this data by mapping the available fields to the Health Check input interface. To do this, use the following procedure:

1. Open Director.
2. Create a new Data Store that points at the data.
3. Create a new snapshot using this Data Store as the source.
4. Add and configure a new mapping to the relevant Data Interface (**Entity Data** or **Individual Data**).
5. Edit the relevant job (**Run Entity** or **Run Individual Data Quality Health Check**), adding the new Snapshot and selecting the new Data Interface mapping.

For full details on how to prepare data, see the following topics in the *Oracle Enterprise Data Quality Director Online Help*:

- "Connecting to a Data Store"
- "Adding a Snapshot "
- "Managing Data Interfaces"
- "Running Jobs using Data Interfaces"

4.5.2 Viewing Data Quality Health Check Results

Health Check results can be produced as four output types:

- Business Intelligence (BI) output;
- EDQ Dashboard results;
- Server Console results; and
- Results Books in EDQ.

4.5.2.1 BI Output

Health Check can produce two comma-separated files containing Individual and Entity results data. This output is intended for detailed analysis using an external Business Intelligence application.

The files are:

- `entityoutput.csv`
- `individualoutput.csv`

Records passed into Health Check will cause one or more rows to be generated, depending on the content of each record and how many errors are discovered within each record.

Note : The separators, and file names and locations within the landing area can be configured in the Run Profile.

The most important metadata attributes in the `.csv` files are as follows:

Column	Description
`entityid / individualid`	The id of the original record.
`Data Stream`	This field identifies the origin of the row:`Misc Data` - A record fields.`AddressData` - An address.`AltPhoneData` - An `altphone` field.
`Rule ID`	The ID of the rule triggered, if applicable.
`Rule Label`	The label of the rule triggered, if applicable.
`Error Code`	The code of the error, if applicable.
`Error Severity`	The severity level of the error, if applicable.
`Error Message`	The error message returned, if applicable.

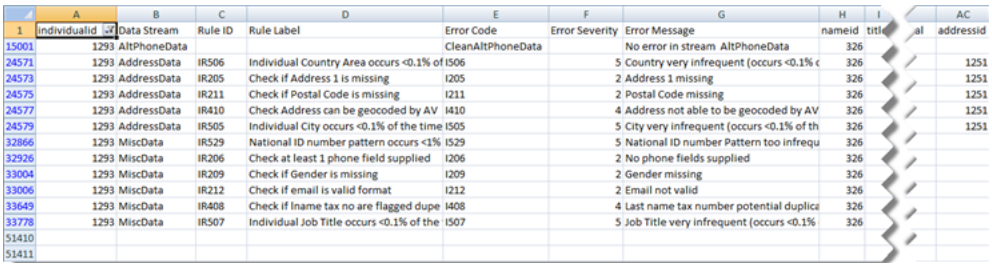
The logic is as follows:

- Each record passed into Health Check returns at least one row in the corresponding `.csv` file.
- At least one row is generated per record. If there is an error in the record data, this is indicated in the Error Code, Error Severity and Error Message columns.
- An additional row is generated per address or `altphone` field within each record. Again, if there is a single error in an address or `altphone` field, this is indicated in the Error columns.
- However, if a record, address or `altphone` field contains more than one error, then a row is generated for each additional error above one.

For example, if an individual record has:

- no address or `altphone` value and no errors: 1 row.
- no address or `altphone` value, and one error: 1 row.
- no address or `altphone` value, and two errors: 2 rows.
- an address, but no `altphone`: 2 rows.
- an address and an `altphone`: 3 rows.
- an address containing a single error, and an `altphone`: 3 rows.
- an address containing two errors, and an `altphone`: 4 rows.

The following is a complex example. The record with `individualid 1293` has returned 12 rows:



	A	B	C	D	E	F	G	H	I	AC
1	individualid	Data Stream	Rule ID	Rule Label	Error Code	Error Severity	Error Message	nameid	title	addressid
15001	1293	AltPhoneData			CleanAltPhoneData		No error in stream AltPhoneData	326		
24571	1293	AddressData	IR506	Individual Country Area occurs <0.1% of	I506	5	Country very infrequent (occurs <0.1%	326		1251
24573	1293	AddressData	IR205	Check if Address 1 is missing	I205	2	Address 1 missing	326		1251
24575	1293	AddressData	IR211	Check if Postal Code is missing	I211	2	Postal Code missing	326		1251
24577	1293	AddressData	IR410	Check Address can be geocoded by AV	I410	4	Address not able to be geocoded by AV	326		1251
24579	1293	AddressData	IR505	Individual City occurs <0.1% of the time	I505	5	City very infrequent (occurs <0.1% of th	326		1251
32866	1293	MiscData	IR529	National ID number pattern occurs <1%	I529	5	National ID number Pattern too infrequ	326		
32926	1293	MiscData	IR206	Check at least 1 phone field supplied	I206	2	No phone fields supplied	326		
33004	1293	MiscData	IR209	Check if Gender is missing	I209	2	Gender missing	326		
33006	1293	MiscData	IR212	Check if email is valid format	I212	2	Email not valid	326		
33649	1293	MiscData	IR408	Check if lname tax no are flagged dupe	I408	4	Last name tax number potential duplica	326		
33778	1293	MiscData	IR507	Individual Job Title occurs <0.1% of the	I507	5	Job Title very infrequent (occurs <0.1%	326		
51410										
51411										

It has the following:

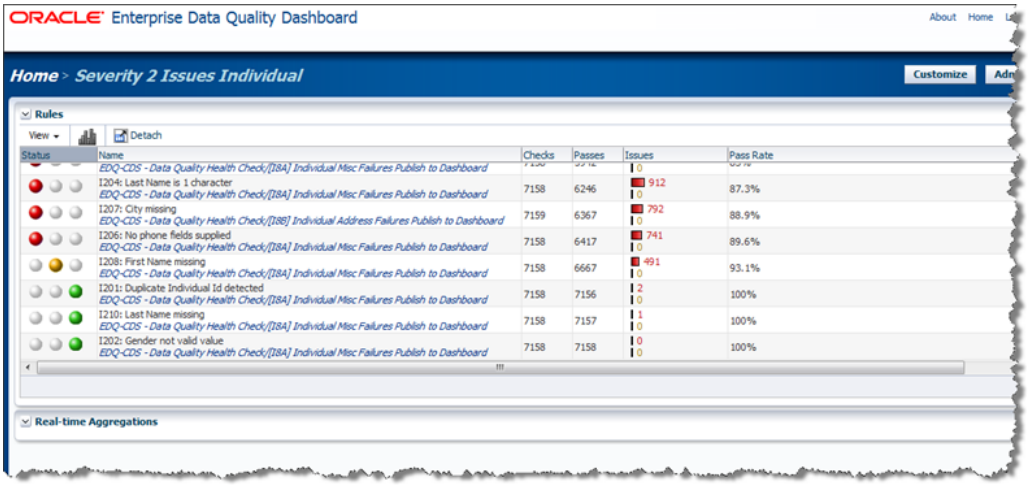
- One `altphone` field, free of errors.

- Five errors associated with one address.
- Six errors associated with other fields in the record (for example, Misc Data.)

> **Note :** In the example file, the `addressid` in each row is identical, which shows that only one address is associated with the record. The illustration does not show this because of the limit of the screen size.

4.5.2.2 EDQ Dashboard

The results published to the Dashboard are dependent on the enabled Business Rules, see Section 4.6, "Managing Business Rules". The following Dashboard example illustrates the variations of results and statuses:



The results from attributes associated with the Individual or Entity record (such as, `name`, `title`, `email` and so on) are based on distinct Individual and Entity records identified by a unique record ID.

Checks on the `altphone` attribute and address-related attributes are performed separately so that the number of results produced correctly reflects the number of child entities processed.

Similarly, results from the `altphone` field are based on distinct alternate phone numbers in Individual and Entity records, as it is possible to have multiple `altphone` values per record.

The results from attributes associated with addresses (such as, `city`, `postalcode`, `country` and so on) are based on distinct address records identified by a unique address id because it is possible to process multiple addresses for a given Individual or Entity.

The number of checks for a given published rule in the Dashboard may vary depending on the type of data being checked, and will always relate to the total population of the type of data. So the "total" figures displayed may vary according to data type.

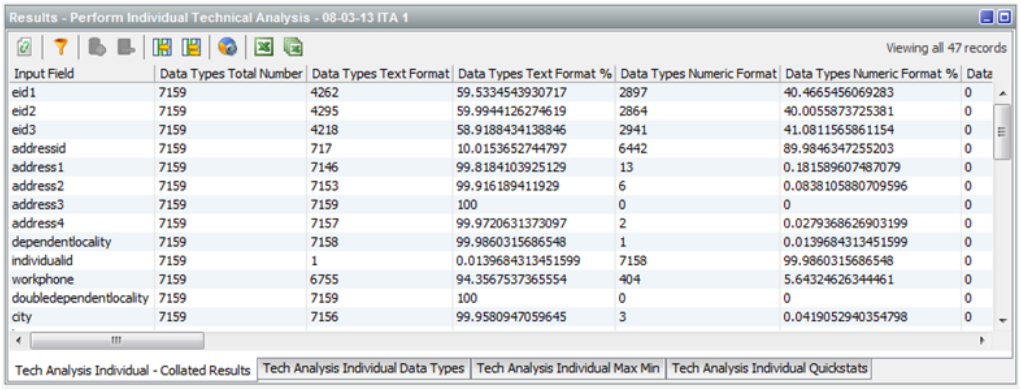
For example, if 500,000 records were passed from the customer system, with a total of 650,000 addresses attached, and a total of 550,000 alternate phone numbers associated with them, then all results will show:

- all address-related rule failures/passes as a percentage of 650,000;

- all alternate-phone-related rule failures/passes as a percentage of 550,000; and
- all remaining rule failure/passes as a percentage of 500,000.

4.5.2.3 Server Console

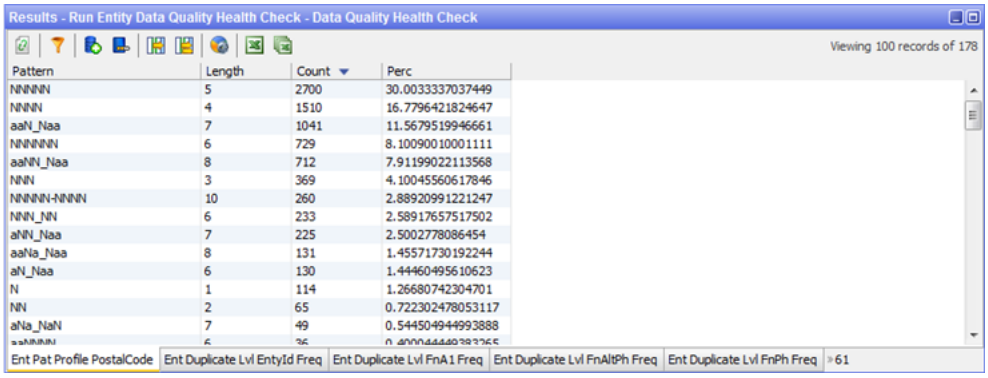
When run in Server Console, the Technical Analysis jobs profile the data by data type, maximum and minimum values and quick stats:



The Health Check jobs perform audit checks on the data and populate the EDQ Dashboard and BI `.csv` files depending on your run profile configuration.

> **Note :** Running the jobs in Server Console does *not* populate the Health Check Results Books.

An example of the Server Console Results, depending on the Run Profile, is as follows:



4.5.2.4 Results Books

If activated in the Health Check Run Profile, the following Results Books can be populated:

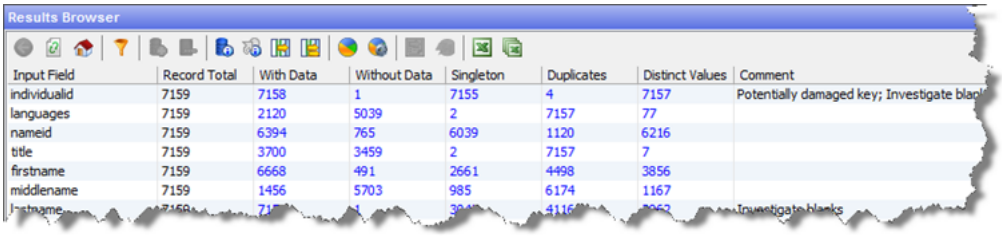
- Entity Profiling Results
- Entity Rules Results
- Entity Technical Analysis
- Individual Profiling Results
- Individual Rules Results
- Individual Technical Analysis

The Technical Analysis Results Books are populated by the corresponding Technical Analysis jobs. The Profiling Results and Rules Results Books are populated by the corresponding Health Check jobs.

Consider the following:

- When running these jobs, select the **edq-cds-data-quality-health-check** Run Profile, but *do not* specify a Run Label.
- The Results Books are *only* populated if the Data Quality jobs are run from EDQ. Running the jobs either from Siebel or Server Console will not populate Results Book data.
- The Business Object grouping of rules in Results Books is pulled from the `Business Object` column in `rulesreference.xls` where each rule is associated with a business object text value. To reclassify rules, edit the `Business Object` column.
- The Technical Analysis jobs only use customer data and publish the analysis results to Server Console or in Results Books only.

It is possible to drill-down through these results for further analysis. Drillable results are links (highlighted in blue):



4.6 Managing Business Rules

This section provides several examples describing how to turn on, edit and add business rules.

4.6.1 Example - Turning on a Rule

The Entity rule `ER418 - Country is missing` is disabled by default.

To turn the rule on:

1. Navigate to the **`oedq_local_home/landingarea/dqhealthcheck`** folder.
2. Open the **`rulesreference.xls`** file.
3. Select the **Address** tab.
4. Find the E418 rule row, and change the value of the cell in the **Enabled** column to **yes**.
5. Save the file.
6. If required, open the Dashboard Administration application to add the rule to an appropriate Summary.

To disable the rule again, repeat this procedure, changing the cell value back to **no**.

Note : If a rule that is included in a Dashboard Summary is disabled, it will still be displayed in the Summary with no results returned. Therefore, it is recommended that any disabled rules be removed from Dashboard Summaries so they do not influence overall pass or failure indicators.

4.6.2 Example - Editing Rules: Adding an Extra Common Title

The `titles` tab in the `dq-health-check-business-rules-individual.xls` file is used by rule `IR411- Check Upper Case Title is in the list`.

The following procedure shows how to ensure the rule also checks for the term "PROFESSOR" as an common title:

1. Navigate to the **`oedq_local_home/businessrules`** folder, and open the **`dq-health-check-business-rules-individual.xls`** spreadsheet.
2. Select the **titles** tab.
3. Add **PROFESSOR** to the bottom of the list in column A of the worksheet.
4. Save the file.

4.6.3 Example - Editing a Rule: Changing a Value Check

This example describes how to change the value check of the `IR430 - Check if DOB is very old (<1900)` to check for birthdates older than 1890.

1. Navigate to the **`oedq_local_home/businessrules`** folder, and open the **`dq-health-check-business-rules-individual.xls`** spreadsheet.
2. Select the **Rules** tab, and scroll to the **IR430** rule.
3. Change the Rule Label to **Check if DOB is very old (<1890)**.
4. Scroll to the **Check1** column, and change the cell value to **chGreaterThan1890**.
5. Select the **Checks** tab, and select the two rows that describe the **chGreaterThan1900** check. In an unmodified sheet, these are normally rows 39 and 40.
6. Copy the rows, and paste them below the existing Checks.
7. Edit the Description, Check Name and Option 1 cells, replacing "1900" with **1890**.
8. Save the file.
9. Open **Director**, and navigate to the **Processe**s node of the EDQ-CDS Data Quality Health Check project in the Project Browser.
10. Double click the **[I8A] Individual Misc Failures Publish to Dashboard** process.
11. In the Process Canvas, locate the following processors in the **Misc Checks** group:
    - IR430 DOB Year is older than 1900 Enabled
    - IR430 DOB Year is older than 1900
12. Edit the labels of these processors (for example, change "1900" to "1890").
13. Double click the **IR430 DOB Year is older than 1890** processor.
14. Select the **Dashboard** tab in the Processor dialog.
15. Edit the rule name to read **I430: DOB year is older than 1890**.

16. Click **OK**.
17. Close the process, saving the changes made.
18. Navigate to the **oedq_local_home/landingarea/dqhealthcheck** folder.
19. Open the **rulesreference.xls** file.
20. Select the **Misc** tab and find the IR430 rule.
21. Change the Description to **DOB year is older than 1890**.
22. Save the file.

4.6.4 Example - Editing a Rule: Changing the Severity Level

This example describes how to change the severity level of rule `IR308 - Check if email is missing` from 3 to 2.

1. Navigate to the **oedq_local_home/businessrules** folder, and open the **dq-health-check-business-rules-individual.xls** spreadsheet.
2. Select the **rules** tab and locate the **IR308** rule.
3. Scroll to the **Error Severity** column and change the cell value to **2**.
4. Save the file.
5. If Severity Summaries have already been configured for Dashboard, open Dashboard Administration, remove the IR308 rule from the Severity 3 Summary and add it to the Severity 2 Summary.

4.6.5 Example - Adding a Rule

This example describes how to add a rule to check that a delivery address post code field passed into the `customstring1` attribute in individual records contains no more than 9 digits, excluding punctuation (for example, conforms to the US zip code format). For this rule to be effective, it will be necessary to clean the field data first by removing any spaces or punctuation marks. This will ensure that only the alphanumeric content is checked

There are eight stages to adding this rule:

1. Confirm the field is passed to the Business Rules processor.
2. Check field format to check the results of a previously-run job in the Server Console Results window, specifically the DQ Health Check Analysis Output tab. This tab is not visible by default. Therefore, before running through this example ensure the `stageddata.Individual\ DQ\ Health\ Check\ Analysis\ Output.visible` attribute is set to `Yes`.
3. Insert pre-processing to reformat the field data.
4. Edit the Business Rules spreadsheet.
5. Edit the **rulesreference.xls** spreadsheet.
6. Change the Business Rules Check processor.
7. Configure for Dashboard.
8. Update the Dashboard Summaries.

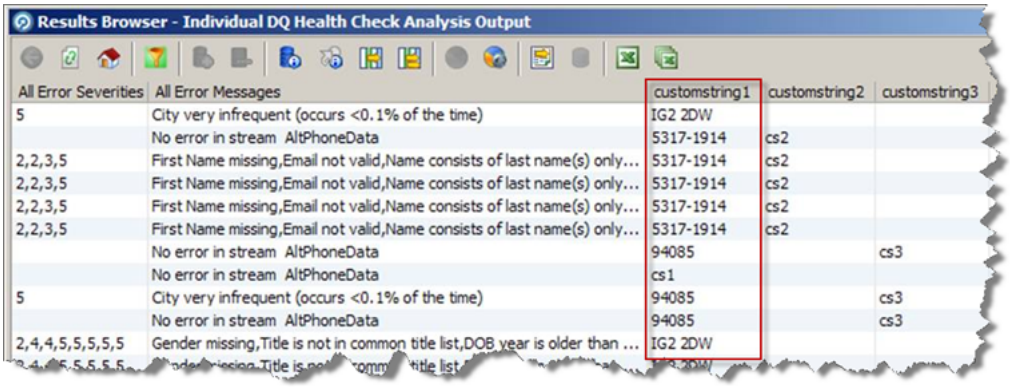
Note: The following examples require a solid understanding of process design in Director and the associated permissions.

Confirming the Field is Passed to the Business Rules Processor

1. Open **Director**, and navigate to the **Processes** node of the **EDQ-CDS - Data Quality Health Check** project in the Project Browser.
2. Double click the **[I6A] Run Misc Business Rules** process.
3. Double-click the **Business Rules Check** processor in the **Business Rules Execution** group at the bottom of the Process Canvas.
4. In the **Attributes** tab of the processor dialog, scroll through the **Attributes** field to confirm the **customstring1** attribute is included.
5. Click the **Identify** tab.
6. Check the Identifier assigned to the **customstring1** Input Attribute (`atCustomString1` in a default installation).

Checking the `field` Format

1. Start Server Console.
2. In the **Results** view, select a previous run of Health Check.
3. The **DQ Health Check Analysis Output** tab should be displayed at the bottom of the window by default. Scroll across to view the **customstring1** column and check the format of the results. In the example image, the format is clearly incorrect: as one field contains a space and the other a hyphen it is not limited to alphanumeric data only:



All Error Severities	All Error Messages	customstring1	customstring2	customstring3
5	City very infrequent (occurs <0.1% of the time)	IG2 2DW		
	No error in stream AltPhoneData	5317-1914	cs2	
2,2,3,5	First Name missing,Email not valid,Name consists of last name(s) only...	5317-1914	cs2	
2,2,3,5	First Name missing,Email not valid,Name consists of last name(s) only...	5317-1914	cs2	
2,2,3,5	First Name missing,Email not valid,Name consists of last name(s) only...	5317-1914	cs2	
2,2,3,5	First Name missing,Email not valid,Name consists of last name(s) only...	5317-1914	cs2	
	No error in stream AltPhoneData	94085		cs3
	No error in stream AltPhoneData	cs1		
5	City very infrequent (occurs <0.1% of the time)	94085		cs3
	No error in stream AltPhoneData	94085		cs3
2,4,4,5,5,5,5,5	Gender missing,Title is not in common title list,DOB year is older than ...	IG2 2DW		

4. Close Server Console.

Inserting Pre-Processing to Format Field Data

As the format of the data in the `customstring1` field does not match the required 9-character alphanumeric format, some pre-processing of the data is required before it is passed to the Business Rules Check processor. Also, to avoid affecting the output of the processor, the pre-processing will be performed on a copy of the `customstring1` data that will then be passed to the check.

1. Return to **Director**.
2. Add a **Concatenate** processor to the **[I16A] Run Misc Business Rules** process, positioning it immediately before the Business Rules Check processor.

3. Configure the processor to take a copy of the customstring1 string, called **customstring1ForChk**.
4. Follow this processor with a **Remove Whitespace** and **Denoise** processor, configuring them to clean the customstring1ForChk data.
5. Save the changes. Leave Director open, as further changes to the Business Rules Check processor are required.

Editing the Business Rules Spreadsheet

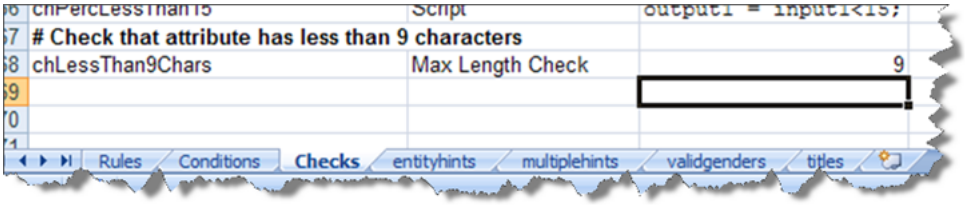
It is now possible to edit the `dq-health-check-business-rules-individual.xls` spreadsheet. This involves adding a new Check, Condition and Business Rule.

Note : The Condition is required in order to ensure that the rule is not applied in circumstances where the `customstring1ForChk` field is not present in the data being analyzed.

---

1. Navigate to the `oedq_local_home/businessrules` folder, and open the `dq-health-check-business-rules-individual.xls` spreadsheet.
2. Click the **Checks** tab.
3. Create a new entry for a check specifying a maximum length of nine characters.

Note : The wording describes the check taking place. In order to fail entries of more than nine characters, the check performed is actually whether the entries are nine characters long or less.



4. Click the **Conditions** tab.
5. Copy and paste the **coCustomString1_supplied** row into an empty row at the bottom of the sheet.
6. Edit the Condition Name and Attribute or Check cells of the new entry to read **coCustomStringForChk_supplied** and **coCustomStringForChk** respectively.
7. Click the **Rules** tab.
8. Add a new line describing the rule, applying the following values:
   - Rule ID: **IR391**
   - Rule Label: **Custom String 1 (denoised) greater than 9 chars**
   - Disable: Leave blank.
   - Apply to Attribute: **atCustomString1ForChk**
   - Condition: **coCustomString1ForChk_supplied**
   - Error Code: **I391**

- Error Severity: **3**
- Error Message: **Custom String 1 (denoised) is greater than 9 characters**
- Check1: **chLessThan9Chars**

9. Click **Save** and close the spreadsheet.

Editing the `rulesreference.xls` Spreadsheet

1. Navigate to the **`oedq_local_home/landingarea/dqhealthcheck`** folder.
2. Open the **`rulesreference.xls`** file.
3. Click the **Misc** tab.
4. Add the details of the new rule to the bottom of the worksheet, as illustrated in following:

	A	B	C	D	E	F	G
1	Rule	Error	Enabled	Description	Type	Business Object	
101	IR528	I528	yes	Work phone Pattern too infrequent (occurs<5% of the time)	Individual	Phone	
102	IR529	I529	yes	National ID number Pattern too infrequent (occurs<1% of the time)	Individual	ID Numbers	
103	IR533	I533	yes	DOB length outside norms (Occurs in top/bottom 0.1%)	Individual	Date of Birth	
104	IR534	I534	yes	Fax phone length outside norms (Occurs in top/bottom 0.1%)	Individual	Phone	
105	IR535	I535	yes	Gender length outside norms (Occurs in top/bottom 0.1%)	Individual	Contact Data	
106	IR536	I536	yes	Home phone length outside norms (Occurs in top/bottom 0.1%)	Individual	Phone	
107	IR537	I537	yes	Mobile phone length outside norms (Occurs in top/bottom 0.1%)	Individual	Phone	
108	IR538	I538	yes	Tax number length outside norms (Occurs in top/bottom 0.1%)	Individual	ID Numbers	
109	IR539	I539	yes	Work phone length outside norms (Occurs in top/bottom 0.1%)	Individual	Phone	
110	IR540	I540	yes	National ID number length outside norms (Occurs in top/bottom 0.1	Individual	ID Numbers	
111	IR391	I391	yes	Custom String 1(denoised) greater than 9 characters	Individual	Contact Data	
112							

5. Click **Save** and close the spreadsheet.

Changing the Business Rules Check Processor

The Business Rules Check processor must be changed to use the reformatted field:

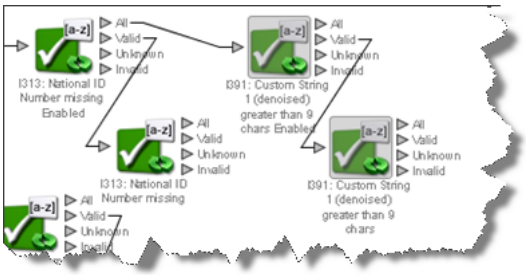
1. Return to **Director**.
2. Double-click the **Business Rules Check** processor in the **[I6A] Run Misc Business Rules** process.
3. On the **Attributes** tab of the **Processor** dialog, add the **customstring1ForChk** attribute to the Input Attributes.
4. Click the **Identify** tab.
5. Find the **atCustomString1ForChk** identifier, and assign the **customstring1ForChk** input attribute in the drop-down field to it.
6. Save the changes and close the dialog.

Configuring for Dashboard

If the Dashboard is used, it is necessary to make further changes to publish the results of the new rule.

1. In **Director**, open the **[I8A] Individual Misc Failures Publish to Dashboard** process.
2. Make a copy of a group of two of the Audit processors. For the purposes of this example, copy the I313 Audit Processors.
3. Paste the copies onto the canvas to the right of the I313 Processors.

4. Rename both copied processors: **I391: Custom String 1 (denoised) greater than 9 chars Enabled** and **I391: Custom String 1 (denoised) greater than 9 chars**.
5. Connect the **All** output of the I313: National ID Number missing processor to the I391: Custom String 1 (denoised) greater than 9 chars Enabled processor. The processors should now appear as in the following:



6. Double-click the **I391: Custom String 1 (denoised) greater than 9 chars Enabled** processor.
7. In the Processor dialog, click the **Options** tab.
8. Set the Regular Expression field to **I391** .
9. Click **Save** and close the dialog.
10. Repeat steps 7 to 9 for the **I391: Custom String 1 (denoised) greater than 9 chars** processor.
11. Click the **Dashboard** tab.
12. Set the **Rule Name** field to "I391: Custom String 1 (denoised) greater than 9 chars".
13. Save changes, and close the process.

Updating the Dashboard Summaries

Once the **Individual Data Quality Health Check** job has been run again, it is possible to add the new rule to the required Summary in the Dashboard Administration application.

4.7 Understanding Data Interfaces

This section describes the two Health Check data interfaces and all of the attributes contained in each of them.

- Section 4.7.1, "Individual Data"
- Section 4.7.2, "Entity Data"

4.7.1 Individual Data

All the Individual Data attributes are strings:

Attribute	Description
`individualid`	Unique identifier of the individual (e.g customer, employee or contact).
`languages`	Three-character Siebel language code. Only used by EDQ-CDS in name standardization to help determine whether a name containing Kanji is Japanese or Chinese.

Attribute	Description
nameid	Unique identifier for the name. Used by EDQ-CDS to distinguish between different names for the same individual when multiple child entities are used. For more information, see Chapter 5, "Using Business Services."
title	
firstname	
middlename	
lastname	
gender	M or F.
dob	Date of Birth in one of the formats listed in the ***Date Formats** EDQ Reference Data set.
jobtitle	
homephone	
workphone	
mobilephone	
faxphone	
alternatephone	
email	
taxnumber	
nationalidnumber	Social Security Number (US) or equivalent.
accountname	The name of the account (for example, entity) to which this individual belongs, if relevant.
uid1	Unique ID 1 **NOTE**: The Unique ID fields are used in EDQ-CDS to match records based on custom unique identifiers, such as passport or tax numbers. For more information, see Chapter 3, "Using Matching."
uid2	Unique ID 2.
uid3	Unique ID 3.
eid1	Elimination ID 1. **Note**: The Elimination ID fields are used in EDQ-CDS to eliminate possible matches between records based on custom unique identifiers, such as passport or tax numbers. For more information, see Chapter 3, "Using Matching."
eid2	Elimination ID 2.
eid3	Elimination ID 3.
addressid	Unique identifier for the address, used in EDQ-CDS to distinguish between different addresses for the same individual when multiple child entities are used. For more information, see Chapter 5, "Using Business Services."
address1	Line 1 of the address.
address2	Line 2 of the address.
address3	Line 3 of the address.
address4	Line 4 of the address.

Attribute	Description
dependentlocality	A smaller population center data element than city, for example, a Turkish neighborhood.
doubledependentlocality	The smallest population center data element, dependent on both the contents of the city and dependentlocality fields. For example, UK Village.
city	
subadminarea	The smallest geographic data element within a country. For example, USA County.
adminarea	The most common geographic data element within a country. For example, USA State or Canadian Province.
postalcode	
country	Country name or ISO 2 char code. **Note**: The output will always be the full Country name, even if the input is the country ISO code.
customstring1	The customstring fields are placeholders for data attributes that require analysis in Health Check but do not match to any of the standard interface attributes.
customstring2	
customstring3	
customstring4	
customstring5	
customstring6	
customstring7	
customstring8	
customstring9	
customstring10	

4.7.2 Entity Data

All the Entity Data attributes are strings.

Attribute	Description
nameid	Unique identifier for the name, used in EDQ-CDS to distinguish between different names for the same entity when multiple child entities are used. For more information, see Chapter 5, "Using Business Services."
entityid	Unique record identifier.
languages	Three-character Siebel language code. Only used in EDQ-CDS for name standardization to help determine whether a name containing Kanji is Japanese or Chinese.
name	Organization name, for example, "Oracle Corporation UK".
subname	Department or site, for example, "Reading" or "Accounts Payable".
phone	
alternatephone	

Attribute	Description
website	
taxnumber	
vatnumber	
uid1	Unique ID 1 **Note**: The Unique ID fields are used in EDQ-CDS to match records based on custom unique identifiers, such as passport or tax numbers. For more information, see Chapter 3, "Using Matching."
uid2	Unique ID 2.
uid3	Unique ID 3.
eid1	Elimination ID 1. **Note**: The Elimination ID fields are used in EDQ-CDS to eliminate possible matches between records based on custom unique identifiers, such as passport or tax numbers.For more information, see Chapter 3, "Using Matching."
eid2	Elimination ID 2.
eid3	Elimination ID 3.
addressid	Unique identifier for the address.
address1	
address2	
address3	
address4	
dependentlocality	A smaller population center data element than city, for example, a Turkish neighborhood.
doubledependentlocality	The smallest population center data element, for example, a UK village.
city	
subadminarea	The smallest geographic data element within a country, for example, US county.
adminarea	The most common geographic data element within a country, for example, US state, Canadian province, UK county.
postalcode	
country	
customstring1	The customstring fields are placeholders for data attributes that require analysis in Health Check but do not match to any of the standard interface attributes.
customstring2	
customstring3	
customstring4	
customstring5	
customstring6	
customstring7	
customstring8	

Attribute	Description
customstring9	
customstring10	

4.8 Dashboard Example Summaries

These tables contain the rules to be included in the Summaries described in Section 4.4.2, "Example - Dashboard By Business Function."

Account - Name Details

Audit Code	Description
E101	Full Name missing
E202	Name is 1 character
E205	Name missing
E302	Sub Name is 1 character
E303	SubName missing
E408	Name contains potential multiples hints
E409	Sub Name contains potential multiples hints
E411	Unusual characters in name
E412	Unusual characters in subname

Account - Identifiers

Audit Code	Description
E102	Entity Id missing
E204	No phone fields supplied
E304	Tax Number missing
E305	VAT Number missing
E306	Website missing
E307	Website not valid
E410	Alternate phone is missing
E413	Unusual characters in phone
E417	Alternate phone is missing
E419	Phone is missing

Account - Identifier Outliers

Audit Code	Description
E420	Alt Phone appears to have less than 2 digits present
E510	Alt phone length outside norms (Occurs in top/bottom 0.1%)
E520	Alt phone Pattern too infrequent (occurs<5% of the time)

Audit Code	Description
E421	Phone appears to have less than 2 digits present
E504	Tax Number too frequent (occurs>5% of the time)
E505	VAT Number too frequent (occurs>5% of the time)
E506	Website too frequent (occurs>5% of the time)
E513	Phone length outside norms (Occurs in top/bottom 0.1%)
E514	Tax number length outside norms (Occurs in top/bottom 0.1%)
E515	VAT number length outside norms (Occurs in top/bottom 0.1%)
E521	Phone Pattern too infrequent (occurs<5% of the time)
E523	Tax number Pattern too infrequent (occurs<1% of the time)
E524	VAT number Pattern too infrequent (occurs<1% of the time)
E525	Website Pattern too frequent (occurs>5% of the time)

Account - Address Details

Audit Code	Description
E203	Address 1 missing
E206	Postal Code missing
E207	City missing
E301	Address not able to be verified by AV processor
E308	Addresses 2 and 3 missing
E407	Address not able to be geocoded by AV processor
E414	Address 2 is missing
E415	Address 3 is missing
E416	Admin area is missing
E418	Country is missing

Account - Address Detail Outliers

Audit Code	Description
E501	Admin Area very infrequent (occurs <0.1% of the time)
E502	City very infrequent (occurs <0.1% of the time)
E503	Country very infrequent (occurs <0.1% of the time)
E511	City length outside norms (Occurs in top/bottom 0.1%)
E512	Country length outside norms (Occurs in top/bottom 0.1%)
E522	Postal code Pattern too infrequent (occurs<1% of the time)

Account - Potential Duplicates

Audit Code	Description
E201	Duplicate Entity Id detected
E401	Full name address1 potential duplicate
E402	Full name alt phone potential duplicate
E403	Full name phone potential duplicate
E404	Full name website potential duplicate
E405	Name tax number potential duplicate
E406	Name VAT number potential duplicate

Contact - Name Details

Audit Code	Description
I101	Full Name missing
I203	First Name is 1 character
I204	Last Name is 1 character
I208	First Name missing
I210	Last Name missing
I301	Name consists of last name(s) only
I304	Middle name is 1 character
I310	Middle name missing
I411	Title is not in common title list
I418	Title is missing
I420	Unusual characters in first name
I421	Unusual characters in last name
I422	Unusual characters in middle name
I428	Full Name contains potential entity hints
I429	Full Name contains potential multiples hints

Contact - Identifiers

Audit Code	Description
I102	Individual Id missing
I206	No phone fields supplied
I212	Email not valid
I302	Gender and title are not consistent
I305	Account Name is missing
I307	DOB missing
I308	Email missing
I311	Tax Number missing

Audit Code	Description
I312	DOB in future
I313	National ID Number missing
I413	Alternate phone is missing
I415	Fax phone is missing
I416	Home phone is missing
I417	Mobile phone is missing
I419	Work phone is missing
I423	Unusual characters in alternate phone
I424	Unusual characters in fax phone
I425	Unusual characters in home phone
I426	Unusual characters in mobile phone
I427	Unusual characters in work phone

Contact - Identifier Outliers

Audit Code	Description
I202	Gender not valid value
I209	Gender missing
I430	DOB year is older than 1900
I433	Alt Phone appears to have less than 2 digits present
I434	Home Phone appears to have less than 2 digits present
I435	Mobile Phone appears to have less than 2 digits present
I436	Work Phone appears to have less than 2 digits present
I437	Fax Phone appears to have less than 2 digits present
I501	Account Name too frequent (occurs>5% of the time)
I502	Email too frequent (occurs>5% of the time)
I503	Tax Number too frequent (occurs>5% of the time)
I508	Title very infrequent (occurs <0.1% of the time)
I509	National ID Number too frequent (occurs>5% of the time)
I510	DOB day in year too frequent (occurs >1% of the time)
I511	DOB Year too frequent (occurs >5% of the time)
I512	DOB Month too frequent (occurs >10% of the time)
I513	DOB Day In Week too frequent (occurs >15% of the time)
I514	DOB Day in Month too frequent (occurs >5% of the time)
I520	Alt phone Pattern too infrequent (occurs<5% of the time)
I521	DOB Pattern too infrequent (occurs<5% of the time)
I522	Email Pattern too frequent (occurs>5% of the time)
I523	Fax phone Pattern too infrequent (occurs<5% of the time)

Audit Code	Description
I524	Home phone Pattern too infrequent (occurs<5% of the time)
I525	Mobile phone Pattern too infrequent (occurs<5% of the time)
I527	Tax number Pattern too infrequent (occurs<1% of the time)
I528	Work phone Pattern too infrequent (occurs<5% of the time)
I529	National ID number Pattern too infrequent (occurs<1% of the time)
I530	Alt phone length outside norms (Occurs in top/bottom 0.1%)
I533	DOB length outside norms (Occurs in top/bottom 0.1%)
I534	Fax phone length outside norms (Occurs in top/bottom 0.1%)
I536	Home phone length outside norms (Occurs in top/bottom 0.1%)
I537	Mobile phone length outside norms (Occurs in top/bottom 0.1%)
I538	Tax number length outside norms (Occurs in top/bottom 0.1%)
I539	Work phone length outside norms (Occurs in top/bottom 0.1%)
I540	National ID number length outside norms (Occurs in top/bottom 0.1%)

Contact - Address Details

Audit Code	Description
I205	Address 1 missing
I207	City missing
I211	Postal Code missing
I303	Address not able to be verified by AV processor
I306	Addresses 2 and 3 missing
I410	Address not able to be geocoded by AV processor
I412	Admin area is missing
I414	Country is missing
I431	Address 2 is missing
I432	Address 3 is missing

Contact - Address Detail Outliers

Audit Code	Description
I504	Admin Area very infrequent (occurs <0.1% of the time)
I505	City very infrequent (occurs <0.1% of the time)
I506	Country very infrequent (occurs <0.1% of the time)
I526	Postal code Pattern too infrequent (occurs<1% of the time)
I531	City length outside norms (Occurs in top/bottom 0.1%)
I532	Country length outside norms (Occurs in top/bottom 0.1%)

Contact - Potential Duplicates

Audit Code	Description
I201	Duplicate Individual Id detected
I401	Full name address1 potential duplicate
I402	Full name alt phone potential duplicate
I403	Full name email potential duplicate
I404	Full name fax phone potential duplicate
I405	Full name home phone potential duplicate
I406	Full name mobile phone potential duplicate
I407	Full name work phone potential duplicate
I408	Last name tax number potential duplicate
I409	Last name national id number potential duplicate

5

Using Business Services

This chapter describes how you can use the EDQ-CDS Business Services functionality.

This chapter includes the following sections:

The provided business services are ready for integration with Siebel Customer Relationship Management (CRM) or Universal Customer Master (UCM) and may also be called by other applications if they are configured to do so.

Ready-to-use, EDQ-CDS provides pre-configured data quality services which can be modified or enhanced by editing the underlying EDQ processes that implement their functionality. These three types of service can be used with Individual, Entity, and Address records:

- Cleaning
- Key Profile (for use in Matching integrations)
- Matching

5.1 Cleaning Services

There are three cleaning services provided in EDQ-CDS: Address Clean, Individual Clean and Entity Clean. The Individual and Entity Clean services are provided as placeholders, which are pre-integrated with Siebel, and easily integrated with other applications, but which need to be modified towards specific requirements.

5.1.1 Address Clean

The EDQ-CDS Address Clean web service provides the following functionality:

- Verification of an input address (returning a verification code and description)
- Geocoding of an address (returning latitude and longitude co-ordinates, with additional metadata)
- Correction, standardization and completion of input addresses (provided the address was verified to a sufficient, configurable, level)

The service is N:N; that is, single and multiple record input and output is possible, but only one record is returned for each record submitted. Each input address is verified and may be corrected, enhanced and geocoded, depending on the options that the job is run with, and the input parameters.

Siebel's Data Quality interface always calls the service with a single record per request, whether running in real-time or batch.

5.1.1.1 Using Address Clean

The Address Clean web service is normally used for real-time verification and cleansing of addresses as they are entered and updated in an application, such as Siebel.

In the case of Siebel, the web service is also used in batch. When a batch address cleansing job is run, the web service will be used on all of the in-scope records in the batch job.

For other applications, it is recommended to add configuration to EDQ-CDS to map data from and to the Address Clean data interface in order to run the service in batch mode.

5.1.1.2 Interface

The following table provides a guide to the interface attributes of the Address Clean web service.

Attribute Name	Data Type	Use	Notes
addressid	String	In/Out	Unique identifier for the address.
address1	String	In/Out	Address line 1
address2	String	In/Out	Address line 2
address3	String	In/Out	Address line 3
address4	String	In/Out	Address line 4
dependentlocality	String	In/Out	A smaller population center data element, dependent on the contents of the city field. For example, a Neighborhood in Turkey. For many countries, this attribute is not used.
doubledependentlocality	String	In/Out	The smallest population center data element, dependent on both the contents of the city and dependentlocality fields. For example, a village in the UK. For many countries, this attribute is not used.
city	String	In/Out	The locality, town or city of the address.
subadminarea	String	In/Out	The smallest geographic data element within a country. For example, a county in the USA.
adminarea	String	In/Out	The most common geographic data element within a country. For example, a State in the USA or a Province in Canada.

Attribute Name	Data Type	Use	Notes
postalcode	String	In/Out	Postal or zip code for the address, if relevant for the country.
country	String	In/Out	On input, an ISO two-character country code (preferred) or country name. On output, the full country name (even if the input is the country ISO code). **Note:** If the country field is blank, the service attempts to derive the country from the city attribute. When this is not present, the default specified in the Run Profile is used.
defaultcountrycode	String	Input only	Default ISO two-character country code to use if country not populated. This overrides the default value used when running the job.
case (parameter)	String	Input only	Transforms output according to the setting: U (Upper)- Transforms all text to Upper case. L (Lower)- Transforms all text to Lower case. M (Mixed) - Transform all text to Mixed case, except postalcode and adminarea. These field values are left as returned from the AV processor if the address was verified. If the address was not verified, the postalcode is left as entered, and the adminarea is converted to Mixed case. O (Original) - Text is not transformed.
mode (parameter)	String	Input only	Mode in which the request is to be run. Currently, only V (Verify) is supported.
minimumverificationmatchscore (parameter)	Number	Input only	A numeric value between 0 and 100, representing the minimum score which a match must achieve to be used as a cleaned address. Input addresses will be left unchanged if the Match Score of the Address Verification processor for the address is lower than the input value.

Attribute Name	Data Type	Use	Notes
minimumverificationlevel (parameter)	Number	Input only	A numeric value between 1 and 5, representing the minimum verification level which a match must achieve to be used as a cleaned address. Input addresses will be left unchanged if the Verification Level of the Address Verification processor for the address is lower than the input value. For a description of what each level means, see Section , "Notes on Verification Levels".
allowedverificationresultcodes (parameter)	String	Input only	A list of any of the following single-letter result codes with no separator (for example, 'VPA'): V (Verified), P (Partially Verified) A (Ambiguous) R (Reverted) U (Unverified) Input addresses will be left unchanged if the Verification Result Code. For example, the first character of the verificationcode) for the address is not one of the listed input values.
fulladdress	String	Output only	Full verified address returned from address verification. The address lines are pipe-separated.
countrycode	String	Output only	ISO 2 char country code of verified address country.
verificationcode	String	Output only	Verification code for the address.
verified	String	Output only	Whether the result produced was verified to a sufficient level, according to the configuration of the process and the resultant verification code. This also indicates whether or not the input address was changed to the returned address from address verification. Possible returned values are Y (verified), N (not verified) or X (EDQ Address Verification is not installed).
verificationcodedescription	String	Output only	US English description of the verification code.
latitude	Number	Output only	WGS 84 latitude in decimal degrees format.
longitude	Number	Output only	WGS 84 longitude in decimal degrees format.

Attribute Name	Data Type	Use	Notes
geoaccuracycode	String	Output only	A code indicating the level of accuracy of the returned geocodes (latitude and longitude) co-ordinates.
geoaccuracycodedescription	String	Output only	US English description of the geoaccuracycode.
geodistance	Number	Output only	Radius of accuracy (in meters) for the returned geocodes. The higher the value, the less accurate the geocoding result.

5.1.1.3 Parameters

The Address Clean web service uses a number of input parameters to control its behavior when processing addresses, as listed and described in the table above.

The minimumverificationmatchscore, minimumverificationlevel and allowedverificationresultcodes parameters are all used as message-level thresholds to override whether or not to change (clean) an input address based on the confidence level that the EDQ Address Verification processor reaches when processing it. Normally, and when using the Address Clean service with Siebel, these parameters are not used, and the underlying settings in the Address Clean process are used to drive whether or not to change the address. In this process it is possible to set these same parameters on a per-country basis if required. Where country-specific thresholds are not provided, global default settings are applied, and these may be set using the EDQ-CDS Run Profile. The priority in which the thresholds are applied is therefore:

1. Per-message threshold settings using the parameter attributes as above
2. Per-country threshold values expressed in the Reference Data set Address Clean - Country verification level and results
3. The global default settings expressed in the process and overridden on a per-run basis by the use of a run profile.

An additional configuration option is available to control the number of address lines that are returned from the service. This is not exposed as a parameter on the interface, but can be set using the phase.*.process.Clean\ -\ Address.Number\ Of\ Address\ Lines Run Profile setting. The default number of lines to return is 4.

Notes on Verification Levels

The following verification levels are possible. The maximum verification level that it is possible to reach varies by country. For information on the maximum level in each country, see the Loqate Oracle EDQ Portal website at

http://www.loqate.com/oracle

The verification level is output as the second character of the Accuracy Code returned by the EDQ Address Verification processor. The 'post-processed' verification level is used (not the 'pre-processed' level); that is, the verification level achieved after EDQ Address Verification applies standardization and parsing to the input address.

Verification Level	Description
1	Verified to Administrative Area (State, Region or County) level
2	Verified to Locality (City or Town) level

Verification Level	Description
3	Verified to Thoroughfare (Street) level
4	Verified to Premise (Building Number) level
5	Verified to Delivery Point (Sub-Building Number) level

Note: If EDQ Address Verification is not installed (or not installed correctly), the Address Clean service can still be installed, and the job that implements it can still be run. However, if a request is made to the service, all the output fields will be blank, except for the `verified` output field, which will have the value *X*, and the `verificationcode` output, which will have the value `-1.0`.

5.1.2 Individual Clean

The Individual Clean web service is designed to verify, correct, standardize or enhance records representing individuals, whether these be customers, prospective customers, contacts, or employees.

The **Clean - Individual** process that implements this service in EDQ-CDS is just a placeholder, and must be customized to requirements. A default process that converts the input name attributes to upper case is provided so that when connecting this service to Siebel or other applications, it is simple to test that the service is correctly connected.

The service is `N:N`; that is, single and multiple record input and output is possible, but only one record is returned for each record submitted.

The Siebel Data Quality interface always calls the service with a single record per request, whether running in real-time or batch.

5.1.2.1 Using Individual Clean

The Individual Clean web service may be extended for many purposes, including (but not limited to):

- Verification of input details related to individuals (for example, an email address)
- Standardization of input details related to individuals (for example, a job title)
- Enhancement of data related to individuals (for example, by matching reference data for individuals and returning additional attributes, such as social media handles)

Normally, the web service will be called in real-time, when individual records are added or updated in an application.

In the case of Siebel, the web service is also used in batch. When a batch contact cleansing job is run, the web service will be used on all of the in-scope records in the batch job.

For other applications, it is recommended to add configuration to EDQ-CDS to map data from and to the **Individual Clean** data interface in order to run the service in batch mode.

The interface used by the service is designed to map directly to the Contact business component in Siebel, but can be freely extended with new attributes on both input and

output. Siebel's DQ vendor parameters may be extended to pass through different attributes to the service.

5.1.2.2 Interface

The following table provides a guide to the default Individual Clean web service interface. All attributes are both input and output by default. It is possible to input an empty value to the interface but to populate the attribute on output, so providing a data enhancement service.

Attribute Name	Data Type	Notes
`individualid`	String	A unique identifier for the individual record.
`languages`	String	3 character Siebel language code. This can be used to determine whether a name containing Kanji should be treated as Japanese or Chinese.
`nameid`	String	Unique identifier for the name.
`title`	String	Title
`firstname`	String	First name
`middlename`	String	Middle name
`lastname`	String	Last Name
`gender`	String	`M` or `F`
`dob`	String	Date of Birth
`jobtitle`	String	Job Title
`homephone`	String	Home Phone Number
`workphone`	String	Work Phone Number
`mobilephone`	String	Mobile Phone Number
`faxphone`	String	Fax Number
`alternatephone`	String	Alternate Phone Number
`email`	String	Email Address
`taxnumber`	String	Tax Number
`nationalidnumber`	String	Social Security Number (US) or equivalent.

5.1.3 Entity Clean

The Entity Clean web service is designed to verify, correct, standardize or enhance records representing entities, whether these be company customers, prospective company customers, suppliers, or other organizations.

The **Clean - Entity** process that implements this service in EDQ-CDS is just a placeholder, and must be customized to requirements. A default process that converts the input name and subname attributes to upper case is provided so that when connecting this service to Siebel or other applications, it is simple to test that the service is correctly connected.

The service is `N:N`; that is, single and multiple record input and output is possible, but only one record is returned for each record submitted.

The Siebel Data Quality interface always calls the service with a single record per request, whether running in real-time or batch.

5.1.3.1 Using Entity Clean

The Entity Clean web service may be extended for many purposes, including (but not limited to):

- Verification of input details related to entities (for example to check that a website is syntactically valid)
- Standardization of input details related to entities (for example company names and locations)
- Enhancement of data related to entities (for example by matching reference data for entities and returning additional attributes, such as DUNS numbers from Dun and Bradstreet)

Normally, the web service will be called in real-time, when entity records are added or updated in an application.

In the case of Siebel, the web service is also used in batch. When a batch account cleansing job is run, the web service will be used on all of the in-scope records in the batch job.

For other applications, it is recommended to add configuration to EDQ-CDS to map data from and to the **Entity Clean** data interface in order to run the service in batch mode.

The interface used by the service is designed to map directly to the Account business component in Siebel, but can be freely extended with new attributes on both input and output. Siebel's Data Quality vendor parameters may be extended to pass through different attributes to the service.

5.1.3.2 Interface

The following table provides a guide to the default Entity Clean web service interface. All attributes are both input and output by default. It is possible to input an empty value to the interface but to populate the attribute on output, so providing a data enhancement service.

Attribute Name	Data Type	Notes
`entityid`	String	A unique identifier for the entity record.
`languages`	String	3 character Siebel language code. This can be used to determine whether a name containing Kanji should be treated as Japanese or Chinese.
`nameid`	String	Unique identifier for the name.
`name`	String	Organization name for example, "Oracle Corporation UK".
`subname`	String	Department or site for example, "Reading", "Accounts Payable", etc.
`phone`	String	Phone Number
`alternatephone`	String	Alternate Phone Number
`website`	String	Website Address
`taxnumber`	String	Company Tax Number
`vatnumber`	String	Company VAT Number

5.2 Key Generation Services

EDQ-CDS key generation services are designed to generate a number of key values for an individual, entity or address record. The returned key values for a record are then used by applications such as Siebel to select 'candidate' records for matching, where any existing record that shares any key value with the 'driving' record (the record submitted to the key generation service) should be considered a candidate. The driving and candidate records are then submitted in a single request to the relevant Matching service.

In addition to being called in real-time in order to prevent the insertion of duplicate records into an application, the key generation services are used in batch mode to populate the key values on all existing records in the application, so that real-time and incremental batch matching jobs, both of which use the key values for existing records for candidate selection, work correctly.

The key generation services are `N:N`; meaning that single and multiple record input and output is possible. In real-time, a single output record is returned containing an array of keys. In batch mode, each key is returned as a separate record in the staging table.

5.2.1 Using Key Generation Services

The real-time key generation services are normally used as the first call to EDQ-CDS when a new or updated record needs to be checked for matches against records in an application. The returned key values are used to select candidate records to be submitted with the driving record to the matching service.

In order to ensure that keys are always up-to-date, the key generation services should be called whenever a record is updated. This includes the scenario when a master record in Siebel UCM, or other hub, is updated due to a confirmed match with an incoming driver record.

The key generation services are exposed using the following:

Web Services:

- `IndividualKeygen`
- `EntityKeygen`
- `AddressKeygen`

Batch Jobs:

- `Batch Individual Key Generation`
- `Batch Entity Key Generation`
- `Batch Address Key Generation`

5.2.2 Interface

The key generation web services present input interfaces with direct mappings to the shared 'Candidate' data interfaces as follows:

Web Service	Input Interface
`IndividualKeygen`	See Section 5.4.1.1, "Individual Candidates."
`EntityKeygen`	See Section 5.4.1.2, "Entity Candidates."

Web Service	Input Interface
AddressKeygen	See Section 5.4.1.3, "Address Candidates."

All the key generation web services return output attributes using the common Real-time Key Generation Results Interface data interface, see Section 5.4.3, "Key Generation Results Interfaces."

The IndividualKeygen, EntityKeygen and AddressKeygen web services all accept a keyprofile attribute in the message header, which is used to specify the enablement of key methods, extending EDQ-CDS to offer a wider menu of key method algorithms.

Due to their verbose nature, automatic profiles are only expected to be generated during Key Analysis and passed in to Key Generation directly from a calling application.

The following out-of-the-box manual profiles are provided with CDS for selection by users:

- Strict
- Typical
- Loose

The actual key methods assigned to these profiles can be seen in the 'Key Profiles - Predefined' reference data in the EDQ-CDS project. Note that they do not necessarily use all the equivalent strict/typical/loose key methods, nor do they exactly correspond to the 1/2/3 legacy cluster levels.

On premise customers can easily add to the list of available key profiles by modifying this reference data accordingly, selecting from the available key methods in the 'Key Methods Master Definition' reference data. New key methods cannot be created without customizing the CDS processes themselves.

5.3 Matching Services

The matching services - sometimes referred to as the Match Scoring services in Siebel Universal Data Quality documentation - compare input (driver and candidate) records and produce a list of possible matches, with scores to indicate how good the matches are, and additional information about how the records matched.

In the matching services, the record for comparison is called a driver, and the records it is compared with are known as candidates. Driver records are also compared with each other, but candidate records are never compared with other candidates. Only the highest scoring match for any given record pair is returned.

Note: Siebel currently does not use the Address Matching service, in either batch or real-time, though this service may be integrated with other applications.

5.3.1 Using Matching Services

There are three forms of matching supported:

Real Time
A single driver record (possibly with multiple child entity records) is compared against many candidates.

Full Batch

All records are compared against one another (subject to key generation); for example, all are specified as drivers. This is an extensive operation that can take some time. It is normally used on a new installation, or perhaps as part of a regular maintenance operation.

Incremental Batch

A specific subset of record types are identified and specified as the driver records. Next, other records with matching key methods are identified, and specified as the candidates. The driver and candidate records are compared, and the driver records are compared with each other. An example of how this might be used is a regular check on all new records during a set time period, such as a week or month, against pre-existing records.

The real-time matching services are exposed via the following web services in EDQ-CDS:

- `IndividualMatch`
- `EntityMatch`
- `AddressMatch`

The batch and real-time processes that implement the matching services use the following Data Interfaces for input data (mapped to the above web services respectively for real-time matching):

- Individual Candidates
- Entity Candidates
- Address Candidates

The **Matches** data interface is used as a common output interface for all record types (Individual, Entity and Address), although the fields mapped for each record type vary.

5.3.2 Matching Using Multiple Identifier Values

Matching services within EDQ-CDS are designed to enable users to submit any number of alternative identifier values to use when matching a given individual or entity; for example, multiple email addresses, addresses or names.

EDQ-CDS can perform matching on multiple values of the following attributes if submitted as pipe-delimited lists:

- uid and eid attributes
- alternatephone
- email
- website
- taxnumber
- nationalidnumber
- vatnumber

However, in order for EDQ-CDS to match Individual or Entity records with multiple names or addresses, such records must first be split into multiple records. Each of these records must have the same `entityid` or `individualid`, but with different `nameid` and/or `addressid` attributes.

So, an Individual record with three names must be split into three records, as follows:

individualid	nameid	firstname	lastname	enail	address1
A1	1	John	Smith	jsmith@jsmith.com	56 High Street
A1	2	Jon	Smith	jsmith@jsmith.com	56 High Street
A1	3	J	Smith	jsmith@jsmith.com	56 High Street

Similarly, an Entity record with two addresses must be split into two records:

entityid	nameid	name	subname	website	address1
B1	1	OracleLtd	Accounts Payable	www.oracle.com	Oracle Parkway
B1	2	Oracle Corporation UK	Accounts Payable	www.oracle.com	Oracle Parkway

And finally, an Entity record with two names and two addresses must be split into four records:

entityid	nameid	name	subname	website	addressid	address1
C1	1	OracleLtd	Accounts Payable	www.oracle.com	A	Oracle Parkway
C1	1	OracleLtd	Accounts Payable	www.oracle.com	B	Thames Valley Park
C1	2	Oracle Corporation UK	Accounts Payable	www.oracle.com	A	Oracle Parkway
C1	2	Oracle Corporation UK	Accounts Payable	www.oracle.com	B	Thames Valley Park

The EDQ Siebel Connector automatically prepares the data to use the matching service appropriately, where EDQ-CDS is integrated with Siebel. If the use of multiple child entities has been configured in Siebel, then the data is prepared in the structure required by the EDQ-CDS matching services. For example, concatenating multiple phone numbers into a pipe-delimited list, and splitting out multiple records if the use of multiple names or addresses is configured.

> **Note:** For records with multiple child entities, only one match will ever be returned between a pair of records. This will always be the highest scoring match according to the match rules.

5.3.3 Interfaces

The matching web services present input interfaces with direct mappings to the shared 'Candidate' data interfaces as follows:

Web Service	Input Interface
`IndividualMatch`	See Section 5.4.1.1, "Individual Candidates."

Web Service	Input Interface
EntityMatch	See Section 5.4.1.2, "Entity Candidates."
AddressMatch	See Section 5.4.1.3, "Address Candidates."

All the matching services return output attributes using the common **Matches Interface** data interface.

5.4 Data Interfaces

This section describes the following EDQ-CDS data interfaces:

- Section 5.4.1, "Candidate Interfaces"
- Section 5.4.2, "Matches Interface"
- Section 5.4.3, "Key Generation Results Interfaces"

5.4.1 Candidate Interfaces

The Candidate interfaces are used for data input to individual/entity/address matching and key generation in both batch and real-time.

> **Note:** In real-time each interface has some associated options which can be passed in on a per-message basis in the header. These all have corresponding options that can be set in the run profile. The header options, if set in real-time, override any values set in the run profile.

5.4.1.1 Individual Candidates

Attribute Name	Data Type	Supports Multiple Values? (Y/N)	Notes
candidate	String	N	0 = driving record, 1 = candidate. Used in matching only. All driving records are compared against each other and against each candidate, but candidates are not compared against each other. Note that in real-time matching, there must be one and only one **logical** driving record representing an Individual, Entity or Address. Multiple **physical** records for the same Individual or Entity are allowed if multiple child entities are in use. Multiple candidate records may always be presented.
individualid	String	N	Unique identifier of the individual (for example, customer, employee, or contact). Mandatory for batch jobs and real time requests containing multiple driver records.
languages	String	Y	3 character Siebel language code. Only used in name standardization to help determine whether a name containing Kanji is Japanese or Chinese.

Attribute Name	Data Type	Supports Multiple Values? (Y/N)	Notes
uid1	String	Y	Unique ID 1 (single or pipe-delimited list of multiple values). **Note**: The Unique ID fields are used to match records based on custom unique identifiers, such as passport or tax numbers. For more information, see *Oracle Enterprise Data Quality Customer Data Services Pack Guide for Enterprise Data Quality*.
uid1id	String	Y	A single or pipe-delimited list of multiple values corresponding to the ID values in uid1, which you can use to identify, in the case of a match, which of the values matched, and return this from the match service.
uid2	String	Y	Unique ID 2 (single or pipe-delimited list of multiple values).
uid2id	String	Y	A single or pipe-delimited list of multiple values corresponding to the ID values in uid2, which you can use to identify, in the case of a match, which of the values matched, and return this from the match service.
uid3	String	Y	Unique ID 3 (single or pipe-delimited list of multiple values).
uid3id	String	Y	A single or pipe-delimited list of multiple values corresponding to the ID values in uid3, which you can use to identify, in the case of a match, which of the values matched, and return this from the match service.
eid1	String	Y	Elimination ID 1 (single or pipe-delimited list of multiple values). **Note**: The Elimination ID fields are used to eliminate possible matches between records based on custom unique identifiers, such as passport or tax numbers. For more information, see *Oracle Enterprise Data Quality Customer Data Services Pack Guide for Enterprise Data Quality*.
eid2	String	Y	Elimination ID 2 (single or pipe-delimited list of multiple values).
eid3	String	Y	Elimination ID 3 (single or pipe-delimited list of multiple values).
ieid1	String	Y	Inverted Elimination ID 1 (single or pipe-delimited list of multiple values).
ieid2	String	Y	Inverted Elimination ID 2 (single or pipe-delimited list of multiple values).
ieid3	String	Y	Inverted Elimination ID 3 (single or pipe-delimited list of multiple values).

Attribute Name	Data Type	Supports Multiple Values? (Y/N)	Notes
nameid	String	N	Unique identifier for the name, used to distinguish between different names for the same individual when multiple child entities are used. For more information, see Section 5.3.1, "Using Matching Services."
title	String	N	Title
firstname	String	N	First Name
middlename	String	N	Middle Name
lastname	String	N	Last Name
gender	String	N	Gender (M or F)
dob	String	N	Date of Birth, in any of the formats recognized by the *Date Formats reference data set in EDQ.
jobtitle	String	N	Job Title
homephone	String	N	Home Phone Number
workphone	String	N	Work Phone Number
mobilephone	String	N	Mobile Phone Number
faxphone	String	N	Fax Number
alternatephone	String	Y	Alternative Phone Number - either a single value, or a pipe-delimited list of multiple values.
alternatephoneid	String	Y	A single or pipe-delimited list of multiple values corresponding to the ID values in alternatephone, which you can use to identify, in the case of a match, which of the values matched, and return this from the match service.
email	String	Y	A single value or a pipe-delimited list of multiple email addresses.
emailid	String	Y	A single or pipe-delimited list of multiple values corresponding to the ID values in email, which you can use to identify, in the case of a match, which of the values matched, and return this from the match service.
taxnumber	String	Y	A single value or a pipe-delimited list of multiple tax numbers.
nationalidnumber	String	Y	Social Security Number (US) or equivalent, single value or pipe-limited list.
accountname	String	N	The name of the account (for example, entity) to which this individual belongs, if relevant.

Attribute Name	Data Type	Supports Multiple Values? (Y/N)	Notes
accountnameid	String	N	An ID field for the accountname field, which you can use to identify, in the case of a match, which account name was matched upon.
addressid	String	N	Unique identifier for the address, used to distinguish between different addresses for the same individual when multiple child entities are used. For more information, see Section 5.3.1, "Using Matching Services."
address1	String	N	Address line 1
address2	String	N	Address line 2
address3	String	N	Address line 3
address4	String	N	Address line 4
dependentlocality	String	N	A smaller population center data element, dependent on the contents of the city field. For example, a Neighborhood in Turkey. For many countries, this attribute is not used.
doubledependentlocality	String	N	The smallest population center data element, dependent on both the contents of the city and dependentlocality fields. For example, a village in the UK. For many countries, this attribute is not used.
city	String	N	The locality, town or city of the address.
subadminarea	String	N	The smallest geographic data element within a country. For example, a county in the USA.
adminarea	String	N	The most common geographic data element within a country. For example, USA State or Canadian Province.
postalcode	String	N	Postal or zip code for the address, if relevant for the country. **Note:** With matching services, leading zeroes are stripped only on numeric postalcodes to avoid a numeric postalcode reinterpreted as a number by an external programs where leading zeroes are automatically stripped. For example, Excel may reformat numeric postalcodes as a number by removing the leading zeroes. This is enabled by default in the edq-cds-daas.properties Run Profile. If there are any alpha characters present, the leading zeroes are not stripped.
country	String	N	Country name or ISO 2 char code.
customstring1	String	N	Custom field for matching on string attribute not provided in interface.

Attribute Name	Data Type	Supports Multiple Values? (Y/N)	Notes
customstring2	String	N	Custom field for matching on string attribute not provided in interface.
customstring3	String	N	Custom field for matching on string attribute not provided in interface.
customstring4	String	N	Custom field for matching on string attribute not provided in interface.
customstring5	String	N	Custom field for matching on string attribute not provided in interface.
customstring6	String	N	Custom field for matching on string attribute not provided in interface.
customdate1	String	N	Custom field for matching on date attribute not provided in interface.
customdate2	String	N	Custom field for matching on date attribute not provided in interface.
customdate3	String	N	Custom field for matching on date attribute not provided in interface.

5.4.1.2 Entity Candidates

Attribute Name	Data Type	Supports Multiple Values? (Y/N)	Notes
candidate	String	N	0 = driving record, 1 = candidate. Used in matching only. All driving records are compared against each other and against each candidate, but candidates are not compared against each other.
entityid	String	N	Unique record identifier. Mandatory for batch jobs and real time requests containing multiple driver records.
languages	String	Y	3 character Siebel language code. Only used in name standardization to help determine whether a name containing Kanji is Japanese or Chinese.
uid1	String	Y	Unique ID 1 (single or pipe-delimited list of multiple values). **Note**: The Unique ID fields are used to match records based on custom unique identifiers, such as passport or tax numbers. For more information, see *Oracle Enterprise Data Quality Customer Data Services Pack Guide for Enterprise Data Quality*.
uid1id	String	Y	A single or pipe-delimited list of multiple values corresponding to the ID values in uid1, which you can use to identify, in the case of a match, which of the values matched, and return this from the match service.

Attribute Name	Data Type	Supports Multiple Values? (Y/N)	Notes
uid2	String	Y	Unique ID 2 (single or pipe-delimited list of multiple values).
uid2id	String	Y	A single or pipe-delimited list of multiple values corresponding to the ID values in uid2, which you can use to identify, in the case of a match, which of the values matched, and return this from the match service.
uid3	String	Y	Unique ID 3 (single or pipe-delimited list of multiple values).
uid3id	String	Y	A single or pipe-delimited list of multiple values corresponding to the ID values in uid3, which you can use to identify, in the case of a match, which of the values matched, and return this from the match service.
eid1	String	Y	Elimination ID 1 (single or pipe-delimited list of multiple values). **Note**: The Elimination ID fields are used to eliminate possible matches between records based on custom unique identifiers, such as passport or tax numbers. For more information, see *Oracle Enterprise Data Quality Customer Data Services Pack Guide for Enterprise Data Quality*.
eid2	String	Y	Elimination ID 2 (single or pipe-delimited list of multiple values).
eid3	String	Y	Elimination ID 3 (single or pipe-delimited list of multiple values).
ieid1	String	Y	Inverted Elimination ID 1 (single or pipe-delimited list of multiple values).
ieid2	String	Y	Inverted Elimination ID 2 (single or pipe-delimited list of multiple values).
ieid3	String	Y	Inverted Elimination ID 3 (single or pipe-delimited list of multiple values).
nameid	String	N	Unique identifier for the name, used to distinguish between different names for the same entity when multiple child entities are used. For more information, see Section 5.3.1, "Using Matching Services."
name	String	N	Organization name, for example, "Oracle Corporation UK".
subname	String	N	Department or site, for example, "Reading" or "Accounts Payable".
phone	String	N	
alternatephone	String	Y	A single or pipe-delimited list of multiple alternative phone number values.

Attribute Name	Data Type	Supports Multiple Values? (Y/N)	Notes
website	String	Y	A single or pipe-delimited list of multiple alternative web site addresses.
taxnumber	String	Y	A single or pipe-delimited list of multiple tax numbers.
vatnumber	String	Y	A single or pipe-delimited list of multiple VAT numbers.
addressid	String	N	Unique identifier for the address, used to distinguish between different addresses for the same entity when multiple child entities are used. For more information, see Section 5.3.1, "Using Matching Services."
address1	String	N	Address line 1
address2	String	N	Address line 2
address3	String	N	Address line 3
address4	String	N	Address line 4
dependentlocality	String	N	A smaller population center data element, dependent on the contents of the city field. For example, Turkish Neighborhood.
doubledependentlocality	String	N	The smallest population center data element, dependent on both the contents of the city and dependentlocality fields. For example, UK Village.
city	String	N	
subadminarea	String	N	The smallest geographic data element within a country. For example, USA County.
adminarea	String	N	The most common geographic data element within a country. For example, USA State or Canadian Province.
postalcode	String	N	Postal or zip code for the address, if relevant for the country. **Note:** With matching services, leading zeroes are stripped only on numeric postalcodes to avoid a numeric postalcode reinterpreted as a number by an external programs where leading zeroes are automatically stripped. For example, Excel may reformat numeric postalcodes as a number by removing the leading zeroes. This is enabled by default in the edq-cds-daas.properties Run Profile. If there are any alpha characters present, the leading zeroes are not stripped.
country	String	N	Country name or ISO 2 char code.
customstring1	String	N	Custom field for matching on string attribute not provided in interface.

Attribute Name	Data Type	Supports Multiple Values? (Y/N)	Notes
customstring2	String	N	Custom field for matching on string attribute not provided in interface.
customstring3	String	N	Custom field for matching on string attribute not provided in interface.
customstring4	String	N	Custom field for matching on string attribute not provided in interface.
customstring5	String	N	Custom field for matching on string attribute not provided in interface.
customstring6	String	N	Custom field for matching on string attribute not provided in interface.
customdate1	String	N	Custom field for matching on date attribute not provided in interface.
customdate2	String	N	Custom field for matching on date attribute not provided in interface.
customdate3	String	N	Custom field for matching on date attribute not provided in interface.

5.4.1.3 Address Candidates

Attribute Name	Data Type	Supports Multiple Values? (Y/N)	Notes
candidate	String	N	0 = driving record, 1 = candidate. Used in matching only.
addressid	String	N	Unique identifier for the address.
address1	String	N	Address line 1
address2	String	N	Address line 2
address3	String	N	Address line 3
address4	String	N	Address line 4
dependentlocality	String	N	A smaller population center data element, dependent on the contents of the city field. For example, Turkish Neighborhood.
doubledependentlocality	String	N	The smallest population center data element, dependent on both the contents of the city and dependentlocality fields. For example, UK Village.
city	String	N	
subadminarea	String	N	The smallest geographic data element within a country. For example, USA County.
adminarea	String	N	The most common geographic data element within a country. For example, USA State or Canadian Province.

Attribute Name	Data Type	Supports Multiple Values? (Y/N)	Notes
postalcode	String	N	Postal or zip code for the address, if relevant for the country.
country	String	N	Country name or ISO 2 char code.

Table 5–1 Individual Candidates - Header Parameters

Name	Type	Default	Description	Run profile param
matchthreshold	Numeric	70	Match threshold (previously on the interface)	phase.Individual\ Match.process.*.matchthreshold
keyprofile	String	Typical	The key profile to use, this is either a named key profile (Typical, etc), or an encoded key profile	phase.*.process.*.keyprofile
overallscore.name.weighting	Positive numeric	7	Weighting for the name compound comparison	phase.Individual\ Match.process.*.overallscore.name.weighting
overallscore.name.enabled	Boolean (Y/N)	Y	Whether to enable the name compound comparison	phase.Individual\ Match.process.*.overallscore.name.enabled
overallscore.address.weighting	Positive numeric	9	Weighting for the address compound comparison	phase.Individual\ Match.process.*.overallscore.address.weighting
overallscore.address.enabled	Boolean (Y/N)	Y	Whether to enable the address compound comparison	phase.Individual\ Match.process.*.overallscore.address.enabled
overallscore.accountname.weighting	Positive numeric	0.75	Weighting for the account name compound comparison	phase.Individual\ Match.process.*.overallscore.accountname.weighting
overallscore.accountname.enabled	Boolean (Y/N)	Y	Whether to enable the account name compound comparison	phase.Individual\ Match.process.*.overallscore.accountname.enabled
overallscore.dob.weighting	Positive numeric	6	Weighting for the date of birth compound comparison	phase.Individual\ Match.process.*.overallscore.dob.weighting
overallscore.dob.enabled	Boolean (Y/N)	Y	Whether to enable the date of birth compound comparison	phase.Individual\ Match.process.*.overallscore.dob.enabled
overallscore.phonenumber.weighting	Positive numeric	5	Weighting for the phone number compound comparison	phase.Individual\ Match.process.*.overallscore.phonenumber.weighting
overallscore.phonenumber.enabled	Boolean (Y/N)	Y	Whether to enable the phone number compound comparison	phase.Individual\ Match.process.*.overallscore.phonenumber.enabled
overallscore.email.weighting	Positive numeric	9	Weighting for the email compound comparison	phase.Individual\ Match.process.*.overallscore.email.weighting
overallscore.email.enabled	Boolean (Y/N)	Y	Whether to enable the email compound comparison	phase.Individual\ Match.process.*.overallscore.email.enabled

Table 5–1 (Cont.) Individual Candidates - Header Parameters

Name	Type	Default	Description	Run profile param
overallscore.nationalidnumber.weighting	Positive numeric	12	Weighting for the nationalid number compound comparison	phase.Individual\ Match.process.*.overallscore.nationalidnumber.weighting
overallscore.nationalidnumber.enabled	Boolean (Y/N)	Y	Whether to enable the nationalid number compound comparison	phase.Individual\ Match.process.*.overallscore.nationalidnumber.enabled
overallscore.taxnumber.weighting	Positive numeric	12	Weighting for the tax number compound comparison	phase.Individual\ Match.process.*.overallscore.taxnumber.weighting
overallscore.taxnumber.enabled	Boolean (Y/N)	Y	Whether to enable the tax number compound comparison	phase.Individual\ Match.process.*.overallscore.taxnumber.enabled
overallscore.customstring1exact.weighting	Positive numeric	1	Weighting for the custom string 1 exact compound comparison	phase.Individual\ Match.process.*.overallscore.customstring1exact.weighting
overallscore.customstring1exact.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 1 exact compound comparison	phase.Individual\ Match.process.*.overallscore.customstring1exact.enabled
overallscore.customstring1fuzzy.weighting	Positive numeric	1	Weighting for the custom string 1 fuzzy compound comparison	phase.Individual\ Match.process.*.overallscore.customstring1fuzzy.weighting
overallscore.customstring1fuzzy.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 1 fuzzy compound comparison	phase.Individual\ Match.process.*.overallscore.customstring1fuzzy.enabled
overallscore.customstring2exact.weighting	Positive numeric	1	Weighting for the custom string 2 exact compound comparison	phase.Individual\ Match.process.*.overallscore.customstring2exact.weighting
overallscore.customstring2exact.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 2 exact compound comparison	phase.Individual\ Match.process.*.overallscore.customstring2exact.enabled
overallscore.customstring2fuzzy.weighting	Positive numeric	1	Weighting for the custom string 2 fuzzy compound comparison	phase.Individual\ Match.process.*.overallscore.customstring2fuzzy.weighting
overallscore.customstring2fuzzy.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 2 fuzzy compound comparison	phase.Individual\ Match.process.*.overallscore.customstring2fuzzy.enabled
overallscore.customstring3exact.weighting	Positive numeric	1	Weighting for the custom string 3 exact compound comparison	phase.Individual\ Match.process.*.overallscore.customstring3exact.weighting
overallscore.customstring3exact.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 3 exact compound comparison	phase.Individual\ Match.process.*.overallscore.customstring3exact.enabled
overallscore.customstring3fuzzy.weighting	Positive numeric	1	Weighting for the custom string 3 fuzzy compound comparison	phase.Individual\ Match.process.*.overallscore.customstring3fuzzy.weighting
overallscore.customstring3fuzzy.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 3 fuzzy compound comparison	phase.Individual\ Match.process.*.overallscore.customstring3fuzzy.enabled
overallscore.customstring4exact.weighting	Positive numeric	1	Weighting for the custom string 4 exact compound comparison	phase.Individual\ Match.process.*.overallscore.customstring4exact.weighting

Table 5–1 (Cont.) Individual Candidates - Header Parameters

Name	Type	Default	Description	Run profile param
overallscore.customstring4exact.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 4 exact compound comparison	phase.Individual\ Match.process.*.overallscore.customstring4exact.enabled
overallscore.customstring4fuzzy.weighting	Positive numeric	1	Weighting for the custom string 4 fuzzy compound comparison	phase.Individual\ Match.process.*.overallscore.customstring4fuzzy.weighting
overallscore.customstring4fuzzy.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 4 fuzzy compound comparison	phase.Individual\ Match.process.*.overallscore.customstring4fuzzy.enabled
overallscore.customstring5exact.weighting	Positive numeric	1	Weighting for the custom string 5 exact compound comparison	phase.Individual\ Match.process.*.overallscore.customstring5exact.weighting
overallscore.customstring5exact.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 5 exact compound comparison	phase.Individual\ Match.process.*.overallscore.customstring5exact.enabled
overallscore.customstring5fuzzy.weighting	Positive numeric	1	Weighting for the custom string 5 fuzzy compound comparison	phase.Individual\ Match.process.*.overallscore.customstring5fuzzy.weighting
overallscore.customstring5fuzzy.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 5 fuzzy compound comparison	phase.Individual\ Match.process.*.overallscore.customstring5fuzzy.enabled
overallscore.customstring6exact.weighting	Positive numeric	1	Weighting for the custom string 6 exact compound comparison	phase.Individual\ Match.process.*.overallscore.customstring6exact.weighting
overallscore.customstring6exact.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 6 exact compound comparison	phase.Individual\ Match.process.*.overallscore.customstring6exact.enabled
overallscore.customstring6fuzzy.weighting	Positive numeric	1	Weighting for the custom string 6 fuzzy compound comparison	phase.Individual\ Match.process.*.overallscore.customstring6fuzzy.weighting
overallscore.customstring6fuzzy.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 6 fuzzy compound comparison	phase.Individual\ Match.process.*.overallscore.customstring6fuzzy.enabled
overallscore.customdate1exact.weighting	Positive numeric	1	Weighting for the custom date 1 exact compound comparison	phase.Individual\ Match.process.*.overallscore.customdate1exact.weighting
overallscore.customdate1exact.enabled	Boolean (Y/N)	Y	Whether to enable the custom date 1 exact compound comparison	phase.Individual\ Match.process.*.overallscore.customdate1exact.enabled
overallscore.customdate1fuzzy.weighting	Positive numeric	1	Weighting for the custom date 1 fuzzy compound comparison	phase.Individual\ Match.process.*.overallscore.customdate1fuzzy.weighting
overallscore.customdate1fuzzy.enabled	Boolean (Y/N)	Y	Whether to enable the custom date 1 fuzzy compound comparison	phase.Individual\ Match.process.*.overallscore.customdate1fuzzy.enabled
overallscore.customdate2exact.weighting	Positive numeric	1	Weighting for the custom date 2 exact compound comparison	phase.Individual\ Match.process.*.overallscore.customdate2exact.weighting
overallscore.customdate2exact.enabled	Boolean (Y/N)	Y	Whether to enable the custom date 2 exact compound comparison	phase.Individual\ Match.process.*.overallscore.customdate2exact.enabled

Table 5–1 (Cont.) Individual Candidates - Header Parameters

Name	Type	Default	Description	Run profile param
overallscore.customdate2fuzzy.weighting	Positive numeric	1	Weighting for the custom date 2 fuzzy compound comparison	phase.Individual\Match.process.*.overallscore.customdate2fuzzy.weighting
overallscore.customdate2fuzzy.enabled	Boolean (Y/N)	Y	Whether to enable the custom date 2 fuzzy compound comparison	phase.Individual\Match.process.*.overallscore.customdate2fuzzy.enabled
overallscore.customdate3exact.weighting	Positive numeric	1	Weighting for the custom date 3 exact compound comparison	phase.Individual\Match.process.*.overallscore.customdate3exact.weighting
overallscore.customdate3exact.enabled	Boolean (Y/N)	Y	Whether to enable the custom date 3 exact compound comparison	phase.Individual\Match.process.*.overallscore.customdate3exact.enabled
overallscore.customdate3fuzzy.weighting	Positive numeric	1	Weighting for the custom date 3 fuzzy compound comparison	phase.Individual\Match.process.*.overallscore.customdate3fuzzy.weighting
overallscore.customdate3fuzzy.enabled	Boolean (Y/N)	Y	Whether to enable the custom date 3 fuzzy compound comparison	phase.Individual\Match.process.*.overallscore.customdate3fuzzy.enabled
clusterlevel	1, 2, 3	2	Legacy clusterlevel (previously on the interface)	phase.*.process.*.clusterlevel
uselegacykeygen	Boolean (Y/N)	N	Whether to use the legacy cluster level setting for key generation	phase.*.process.*.uselegacykeygen
customstring1key	Boolean (Y/N)	N	Whether to key on this custom attribute	phase.*.process.*.customstring1key
customstring1type	String (text/identifier)	text	Type of data contained in the custom attribute - defines whether to strip whitespace when standardizing or not	phase.*.process.*.customstring1type
customstring2key	Boolean (Y/N)	N	Whether to key on this custom attribute	phase.*.process.*.customstring2key
customstring2type	String (text/identifier)	text	Type of data contained in the custom attribute - defines whether to strip whitespace when standardizing or not	phase.*.process.*.customstring2type
customstring3key	Boolean (Y/N)	N	Whether to key on this custom attribute	phase.*.process.*.customstring3key
customstring3type	String (text/identifier)	text	Type of data contained in the custom attribute - defines whether to strip whitespace when standardizing or not	phase.*.process.*.customstring3type
customstring4key	Boolean (Y/N)	N	Whether to key on this custom attribute	phase.*.process.*.customstring4key
customstring4type	String (text/identifier)	text	Type of data contained in the custom attribute - defines whether to strip whitespace when standardizing or not	phase.*.process.*.customstring4type
customstring5key	Boolean (Y/N)	N	Whether to key on this custom attribute	phase.*.process.*.customstring5key
customstring5type	String (text/identifier)	text	Type of data contained in the custom attribute - defines whether to strip whitespace when standardizing or not	phase.*.process.*.customstring5type

Table 5–1 (Cont.) Individual Candidates - Header Parameters

Name	Type	Default	Description	Run profile param
customstring6key	Boolean (Y/N)	N	Whether to key on this custom attribute	phase.*.process.*.customstring6key
customstring6type	String (text/identifier)	text	Type of data contained in the custom attribute - defines whether to strip whitespace when standardizing or not	phase.*.process.*.customstring6type
customdate1key	Boolean (Y/N)	N	Whether to key on this custom attribute	phase.*.process.*.customdate1key
customdate2key	Boolean (Y/N)	N	Whether to key on this custom attribute	phase.*.process.*.customdate2key
customdate3key	Boolean (Y/N)	N	Whether to key on this custom attribute	phase.*.process.*.customdate3key

Table 5–2 Entity Candidates - Header Parameters

Name	Type	Default	Description	Runprofile param
matchthreshold	Numeric	70	Match threshold	phase.Entity\Match.process.*.matchthreshold
keyprofile	String	Typical	The key profile to use, this is either a named key profile (Typical, etc), or an encoded key profile	phase.*.process.*.keyprofile
overallscore.name.weighting	Positive numeric	10	Weighting for the name compound comparison	phase.Entity\Match.process.*.overallscore.name.weighting
overallscore.name.enabled	Boolean (Y/N)	Y	Whether to enable the name compound comparison	phase.Entity\Match.process.*.overallscore.name.enabled
overallscore.address.weighting	Positive numeric	4	Weighting for the address compound comparison	phase.Entity\Match.process.*.overallscore.address.weighting
overallscore.address.enabled	Boolean (Y/N)	Y	Whether to enable the address compound comparison	phase.Entity\Match.process.*.overallscore.address.enabled
overallscore.website.weighting	Positive numeric	0.3	Weighting for the website compound comparison	phase.Entity\Match.process.*.overallscore.website.weighting
overallscore.website.enabled	Boolean (Y/N)	Y	Whether to enable the website compound comparison	phase.Entity\Match.process.*.overallscore.website.enabled
overallscore.phonenumber.weighting	Positive numeric	0.5	Weighting for the phone number compound comparison	phase.Entity\Match.process.*.overallscore.phonenumber.weighting
overallscore.phonenumber.enabled	Boolean (Y/N)	Y	Whether to enable the phone number compound comparison	phase.Entity\Match.process.*.overallscore.phonenumber.enabled
overallscore.taxnumber.weighting	Positive numeric	1	Weighting for the tax number compound comparison	phase.Entity\Match.process.*.overallscore.taxnumber.weighting

Table 5–2 (Cont.) Entity Candidates - Header Parameters

Name	Type	Default	Description	Runprofile param
overallscore.taxnumber.enabled	Boolean (Y/N)	Y	Whether to enable the tax number compound comparison	phase.Entity\ Match.process.*.overallscore.taxnumber.enabled
overallscore.vatnumber.weighting	Positive numeric	1	Weighting for the vat number compound comparison	phase.Entity\ Match.process.*.overallscore.vatnumber.weighting
overallscore.vatnumber.enabled	Boolean (Y/N)	Y	Whether to enable the vat number compound comparison	phase.Entity\ Match.process.*.overallscore.vatnumber.enabled
overallscore.customstring1exact.weighting	Positive numeric	1	Weighting for the custom string 1 exact compound comparison	phase.Entity\ Match.process.*.overallscore.customstring1exact.weighting
overallscore.customstring1exact.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 1 exact compound comparison	phase.Entity\ Match.process.*.overallscore.customstring1exact.enabled
overallscore.customstring1fuzzy.weighting	Positive numeric	1	Weighting for the custom string 1 fuzzy compound comparison	phase.Entity\ Match.process.*.overallscore.customstring1fuzzy.weighting
overallscore.customstring1fuzzy.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 1 fuzzy compound comparison	phase.Entity\ Match.process.*.overallscore.customstring1fuzzy.enabled
overallscore.customstring2exact.weighting	Positive numeric	1	Weighting for the custom string 2 exact compound comparison	phase.Entity\ Match.process.*.overallscore.customstring2exact.weighting
overallscore.customstring2exact.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 2 exact compound comparison	phase.Entity\ Match.process.*.overallscore.customstring2exact.enabled
overallscore.customstring2fuzzy.weighting	Positive numeric	1	Weighting for the custom string 2 fuzzy compound comparison	phase.Entity\ Match.process.*.overallscore.customstring2fuzzy.weighting
overallscore.customstring2fuzzy.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 2 fuzzy compound comparison	phase.Entity\ Match.process.*.overallscore.customstring2fuzzy.enabled
overallscore.customstring3exact.weighting	Positive numeric	1	Weighting for the custom string 3 exact compound comparison	phase.Entity\ Match.process.*.overallscore.customstring3exact.weighting
overallscore.customstring3exact.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 3 exact compound comparison	phase.Entity\ Match.process.*.overallscore.customstring3exact.enabled
overallscore.customstring3fuzzy.weighting	Positive numeric	1	Weighting for the custom string 3 fuzzy compound comparison	phase.Entity\ Match.process.*.overallscore.customstring3fuzzy.weighting
overallscore.customstring3fuzzy.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 3 fuzzy compound comparison	phase.Entity\ Match.process.*.overallscore.customstring3fuzzy.enabled
overallscore.customstring4exact.weighting	Positive numeric	1	Weighting for the custom string 4 exact compound comparison	phase.Entity\ Match.process.*.overallscore.customstring4exact.weighting
overallscore.customstring4exact.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 4 exact compound comparison	phase.Entity\ Match.process.*.overallscore.customstring4exact.enabled

Table 5–2 (Cont.) Entity Candidates - Header Parameters

Name	Type	Default	Description	Runprofile param
overallscore.customstring4fuzzy.weighting	Positive numeric	1	Weighting for the custom string 4 fuzzy compound comparison	phase.Entity\ Match.process.*.overallscore.customstring4fuzzy.weighting
overallscore.customstring4fuzzy.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 4 fuzzy compound comparison	phase.Entity\ Match.process.*.overallscore.customstring4fuzzy.enabled
overallscore.customstring5exact.weighting	Positive numeric	1	Weighting for the custom string 5 exact compound comparison	phase.Entity\ Match.process.*.overallscore.customstring5exact.weighting
overallscore.customstring5exact.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 5 exact compound comparison	phase.Entity\ Match.process.*.overallscore.customstring5exact.enabled
overallscore.customstring5fuzzy.weighting	Positive numeric	1	Weighting for the custom string 5 fuzzy compound comparison	phase.Entity\ Match.process.*.overallscore.customstring5fuzzy.weighting
overallscore.customstring5fuzzy.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 5 fuzzy compound comparison	phase.Entity\ Match.process.*.overallscore.customstring5fuzzy.enabled
overallscore.customstring6exact.weighting	Positive numeric	1	Weighting for the custom string 6 exact compound comparison	phase.Entity\ Match.process.*.overallscore.customstring6exact.weighting
overallscore.customstring6exact.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 6 exact compound comparison	phase.Entity\ Match.process.*.overallscore.customstring6exact.enabled
overallscore.customstring6fuzzy.weighting	Positive numeric	1	Weighting for the custom string 6 fuzzy compound comparison	phase.Entity\ Match.process.*.overallscore.customstring6fuzzy.weighting
overallscore.customstring6fuzzy.enabled	Boolean (Y/N)	Y	Whether to enable the custom string 6 fuzzy compound comparison	phase.Entity\ Match.process.*.overallscore.customstring6fuzzy.enabled
overallscore.customdate1exact.weighting	Positive numeric	1	Weighting for the custom date 1 exact compound comparison	phase.Entity\ Match.process.*.overallscore.customdate1exact.weighting
overallscore.customdate1exact.enabled	Boolean (Y/N)	Y	Whether to enable the custom date 1 exact compound comparison	phase.Entity\ Match.process.*.overallscore.customdate1exact.enabled
overallscore.customdate1fuzzy.weighting	Positive numeric	1	Weighting for the custom date 1 fuzzy compound comparison	phase.Entity\ Match.process.*.overallscore.customdate1fuzzy.weighting
overallscore.customdate1fuzzy.enabled	Boolean (Y/N)	Y	Whether to enable the custom date 1 fuzzy compound comparison	phase.Entity\ Match.process.*.overallscore.customdate1fuzzy.enabled
overallscore.customdate2exact.weighting	Positive numeric	1	Weighting for the custom date 2 exact compound comparison	phase.Entity\ Match.process.*.overallscore.customdate2exact.weighting
overallscore.customdate2exact.enabled	Boolean (Y/N)	Y	Whether to enable the custom date 2 exact compound comparison	phase.Entity\ Match.process.*.overallscore.customdate2exact.enabled
overallscore.customdate2fuzzy.weighting	Positive numeric	1	Weighting for the custom date 2 fuzzy compound comparison	phase.Entity\ Match.process.*.overallscore.customdate2fuzzy.weighting

Table 5–2 (Cont.) Entity Candidates - Header Parameters

Name	Type	Default	Description	Runprofile param
overallscore.customdate2fuzzy.enabled	Boolean (Y/N)	Y	Whether to enable the custom date 2 fuzzy compound comparison	phase.Entity\Match.process.*.overallscore.customdate2fuzzy.enabled
overallscore.customdate3exact.weighting	Positive numeric	1	Weighting for the custom date 3 exact compound comparison	phase.Entity\Match.process.*.overallscore.customdate3exact.weighting
overallscore.customdate3exact.enabled	Boolean (Y/N)	Y	Whether to enable the custom date 3 exact compound comparison	phase.Entity\Match.process.*.overallscore.customdate3exact.enabled
overallscore.customdate3fuzzy.weighting	Positive numeric	1	Weighting for the custom date 3 fuzzy compound comparison	phase.Entity\Match.process.*.overallscore.customdate3fuzzy.weighting
overallscore.customdate3fuzzy.enabled	Boolean (Y/N)	Y	Whether to enable the custom date 3 fuzzy compound comparison	phase.Entity\Match.process.*.overallscore.customdate3fuzzy.enabled
clusterlevel	1, 2, 3	2	Legacy clusterlevel	phase.*.process.*.clusterlevel
uselegacykeygen	Boolean (Y/N)	N	Whether to use the legacy cluster level setting for key generation	phase.*.process.*.uselegacykeygen
customstring1key	Boolean (Y/N)	N	Whether to key on this custom attribute	phase.*.process.*.customstring1key
customstring1type	String (text/identifier)	text	Type of data contained in the custom attribute - defines whether to strip whitespace when standardizing or not	phase.*.process.*.customstring1type
customstring2key	Boolean (Y/N)	N	Whether to key on this custom attribute	phase.*.process.*.customstring2key
customstring2type	String (text/identifier)	text	Type of data contained in the custom attribute - defines whether to strip whitespace when standardizing or not	phase.*.process.*.customstring2type
customstring3key	Boolean (Y/N)	N	Whether to key on this custom attribute	phase.*.process.*.customstring3key
customstring3type	String (text/identifier)	text	Type of data contained in the custom attribute - defines whether to strip whitespace when standardizing or not	phase.*.process.*.customstring3type
customstring4key	Boolean (Y/N)	N	Whether to key on this custom attribute	phase.*.process.*.customstring4key
customstring4type	String (text/identifier)	text	Type of data contained in the custom attribute - defines whether to strip whitespace when standardizing or not	phase.*.process.*.customstring4type
customstring5key	Boolean (Y/N)	N	Whether to key on this custom attribute	phase.*.process.*.customstring5key
customstring5type	String (text/identifier)	text	Type of data contained in the custom attribute - defines whether to strip whitespace when standardizing or not	phase.*.process.*.customstring5type
customstring6key	Boolean (Y/N)	N	Whether to key on this custom attribute	phase.*.process.*.customstring6key

Table 5–2 (Cont.) Entity Candidates - Header Parameters

Name	Type	Default	Description	Runprofile param
customstring6type	String (text/identifier)	text	Type of data contained in the custom attribute - defines whether to strip whitespace when standardizing or not	phase.*.process.*.customstring6type
customdate1key	Boolean (Y/N)	N	Whether to key on this custom attribute	phase.*.process.*.customdate1key
customdate2key	Boolean (Y/N)	N	Whether to key on this custom attribute	phase.*.process.*.customdate2key
customdate3key	Boolean (Y/N)	N	Whether to key on this custom attribute	phase.*.process.*.customdate3key

Table 5–3 Address Candidates - Header Parameters

Name	Type	Default	Description	Run profile param
matchthreshold	Numeric	70	Match threshold	phase.Address\Match.process.*.matchthreshold
keyprofile	String	Typical	The key profile to use, this is either a named key profile (Typical, etc), or an encoded key profile	phase.*.process.*.keyprofile
clusterlevel	1, 2, 3	2	Legacy clusterlevel	phase.*.process.*.clusterlevel
uselegacykeygen	Boolean (Y/N)	N	Whether to use the legacy cluster level setting for key generation	phase.*.process.*.uselegacykeygen

5.4.2 Matches Interface

The Matches interface is used for the output of the matching services in batch and real-time. It is used for individuals, entities and addresses because it contains no attributes specific to any business object.

With Individual and Entity matching, if there are multiple matches between records with the same `masterid` and `matchids` (for example, due to multiple matches with different names and addresses), only the strongest match (by match score) is returned for the record pair. Siebel does not currently use the returned `masternameid`, `matchnameid`, `masteraddressid` and `matchaddressid` attributes, though these may be used in other integrations to display the correct records in the application according to the best matches.

Attribute Name	Data Type	Notes
`serverid`	String	Server ID. Not applicable to Siebel.
`jobid`	String	Job ID. Not applicable to Siebel.
`masterid`	String	Driving record ID. Only used in Batch.
`matchid`	String	Matching record ID.
`masternameid`	String	Driving record name ID. Used to identify which name matched on the driving record, where multiple names were presented.

Attribute Name	Data Type	Notes
matchnameid	String	Matching record name ID. Used to identify which name matched on the candidate record, where multiple names were presented.
masteraccountnameid	String	Driving record account name ID. Used to identify which account name matched on the driving record, where multiple account names were presented.
matchaccountnameid	String	Matching record account name ID. Used to identify which account name matched on the candidate record, where multiple account names were presented.
masteraddressid	String	Driving record address ID. Used to identify which address matched on the driving record, where multiple addresses were presented.
matchaddressid	String	Matching record address ID. Used to identify which address matched on the candidate record, where multiple addresses were presented.
masteremailid	String	Driving record email ID. Used to identify which email address matched on the driving record, where multiple emails were presented.
matchemailid	String	Matching record email ID. Used to identify which email address matched on the candidate record, where multiple emails were presented.
masterphonenumberid	String	Driving record phone number ID. Used to identify which phone number matched on the driving record, where multiple phone numbers were presented.
matchphonenumberid	String	Matching record phone number ID. Used to identify which phone number matched on the candidate record, where multiple phone numbers were presented.
masterwebsiteid	String	Driving record website ID. Used to identify which website matched on the driving record, where multiple websites were presented.
matchwebsiteid	String	Matching record website ID. Used to identify which website matched on the candidate record, where multiple websites were presented.
masteruid1id	String	Driving record unique identifier 1 ID. Used to identify which unique identifier 1 (UID1) value matched on the driving record, where multiple UID1 values were presented.
matchuid1id	String	Matching record unique identifier 1 ID. Used to identify which unique identifier 1 (UID1) value matched on the candidate record, where multiple UID1 values were presented.
masteruid2id	String	Driving record unique identifier 2 ID. Used to identify which unique identifier 2 (UID2) value matched on the driving record, where multiple UID2 values were presented.
matchuid2id	String	Matching record unique identifier 2 ID. Used to identify which unique identifier 2 (UID2) value matched on the candidate record, where multiple UID2 values were presented.
masteruid3id	String	Driving record unique identifier 3 ID. Used to identify which unique identifier 3 (UID3) value matched on the driving record, where multiple UID3 values were presented.

Attribute Name	Data Type	Notes
matchuid3id	String	Matching record unique identifier 3 ID. Used to identify which unique identifier 3 (UID3) value matched on the candidate record, where multiple UID3 values were presented.
matchscore	Number	Match score.
rulename	String	Match rule name. This may be taken from a compound score result if the relationship was generated from a compound score based rule
reversedriverflag	String	A flag indicating that an additional, reversed match record has been generated where there is a match between driving records in Batch matching. Valid values are Y and N.
comparisonresults	String	Comma separated list of attributes contributing to the relationship, and how they matched (for example, Name Exact, Address Fuzzy).
nameresult	String	Result for the name compound comparison.
namescore	Number	Score for the name compound comparison.
namecategory	String	Category (Exact, Fuzzy, etc.) for the name compound comparison.
addressresult	String	Result for the address compound comparison.
addressscore	Number	Score for the address compound comparison.
addresscategory	String	Category (Exact, Fuzzy, etc.) for the address compound comparison.
accountnameresult	String	Result for the accountname compound comparison.
accountnamescore	Number	Score for the accountname compound comparison.
accountnamecategory	String	Category (Exact, Fuzzy, etc.) for the accountname compound comparison.
emailresult	String	Result for the email compound comparison.
emailscore	Number	Score for the email compound comparison.
emailcategory	String	Category (Exact, Fuzzy, etc.) for the email compound comparison.
phonenumberresult	String	Result for the phonenumber compound comparison.
phonenumberscore	Number	Score for the phonenumber compound comparison.
phonenumbercategory	String	Category (Exact, Fuzzy, etc.) for the phonenumber compound comparison.
dobresult	String	Result for the dob compound comparison.
dobscore	Number	Score for the dob compound comparison.
dobcategory	String	Category (Exact, Fuzzy, etc.) for the dob compound comparison.
websiteresult	String	Result for the website compound comparison.
websitescore	Number	Score for the website compound comparison.
websitecategory	String	Category (Exact, Fuzzy, etc.) for the website compound comparison.
nationalidnumberresult	String	Result for the nationalidnumber compound comparison.

Attribute Name	Data Type	Notes
nationalidnumberscore	Number	Score for the nationalidnumber compound comparison.
nationalidnumbercategory	String	Category (Exact, Fuzzy, etc.) for the nationalidnumber compound comparison.
vatnumberresult	String	Result for the vatnumber compound comparison.
vatnumberscore	Number	Score for the vatnumber compound comparison.
vatnumbercategory	String	Category (Exact, Fuzzy, etc.) for the vatnumber compound comparison.
taxnumberresult	String	Result for the taxnumber compound comparison.
taxnumberscore	Number	Score for the taxnumber compound comparison.
taxnumbercategory	String	Category (Exact, Fuzzy, etc.) for the taxnumber compound comparison.
customstring1result	String	Result for the customstring1 compound comparison.
customstring1score	Number	Score for the customstring1 compound comparison.
customstring1category	String	Category (Exact, Fuzzy, etc.) for the customstring1 compound comparison.
customstring2result	String	Result for the customstring2 compound comparison.
customstring2score	Number	Score for the customstring2 compound comparison.
customstring2category	String	Category (Exact, Fuzzy, etc.) for the customstring2 compound comparison.
customstring3result	String	Result for the customstring3 compound comparison.
customstring3score	Number	Score for the customstring3 compound comparison.
customstring3category	String	Category (Exact, Fuzzy, etc.) for the customstring3 compound comparison.
customstring4result	String	Result for the customstring4 compound comparison.
customstring4score	Number	Score for the customstring4 compound comparison.
customstring4category	String	Category (Exact, Fuzzy, etc.) for the customstring4 compound comparison.
customstring5result	String	Result for the customstring5 compound comparison.
customstring5score	Number	Score for the customstring5 compound comparison.
customstring5category	String	Category (Exact, Fuzzy, etc.) for the customstring5 compound comparison.
customstring6result	String	Result for the customstring6 compound comparison.
customstring6score	Number	Score for the customstring6 compound comparison.
customstring6category	String	Category (Exact, Fuzzy, etc.) for the customstring6 compound comparison.
customdate1result	String	Result for the customdate1 compound comparison.
customdate1score	Number	Score for the customdate1 compound comparison.

Attribute Name	Data Type	Notes
customdate1category	String	Category (Exact, Fuzzy, etc.) for the customdate1 compound comparison.
customdate2result	String	Result for the customdate2 compound comparison.
customdate2score	Number	Score for the customdate2 compound comparison.
customdate2category	String	Category (Exact, Fuzzy, etc.) for the customdate2 compound comparison.
customdate3result	String	Result for the customdate3 compound comparison.
customdate3score	Number	Score for the customdate3 compound comparison.
customdate3category	String	Category (Exact, Fuzzy, etc.) for the customdate3 compound comparison.

Note:

- In Siebel integrations, the driving record(s) are also returned in the output from real-time matching requests, with a blank match score and rule name. This behavior is controlled by the `phase.*.process.*.Return\ Real-time\ Driving\ Record` Run Profile property, and therefore could be configured for other types of integration if required.
- So that external applications, such as Siebel, can simply consume the output from batch matching to update both records in a match, CDS batch matching provides two records for each match between driving records. Therefore, if A matches B, a record is returned with `masterid` A and `matchid` B, *and* an additional record is generated and returned with `masterid` B and `matchid` A. This additionally generated record will have `reversedriverflag` set to `Y` in case the external application does not need the additionally generated record.
- The Match Rule Name cannot be displayed in Siebel due to the limitations of the Siebel Data Quality interface that only accepts a returned score related to each matched record.

5.4.3 Key Generation Results Interfaces

Two data interfaces are used for the output of the results of key generation services; one for batch and one for real-time. They are used for entities, individuals, and addresses as they contain no attributes specific to a particular business object. The Batch and Real-Time Results Interfaces contain similar information, the main difference is the way in which the results are processed.

5.4.3.1 Real-Time Key Generation Results Interface

The Real-time Key Generation Results interface is used for the output of key generation services in real-time. The output values are returned in arrays of key values for a record, in no particular order. This interface differs from the batch version in that it returns arrays of key values for a record, rather than a row per key value per record.

Note: If the legacy cluster level is in use, the level will be returned in the key priorities field, unless `clusterlevel` = D in which case it will not be returned (as `keypriorities` is numeric).

Attribute Name	Data Type	Notes
`externalid`	String	ID of the individual, entity or address of the key generated record.
`keyprofile`	String	Key profile of the generated keys.
`keyvalues`	StringArray	Key value array.
`keypriorities`	NumberArray	Key priority array.

5.4.3.2 Batch Key Generation Results Interface

The Batch Key Generation Results interface is only used for the output for key generation services in the Batch Key Generation service. This interface differs from the Real-time Key Generation Results interface in that it returns a row per key value per record, rather than arrays of key values for a record.

Note: If the legacy cluster level is in use, the level will be returned in the key priority field, unless `clusterlevel` = D in which case it will not be returned (as `keypriority` is numeric).

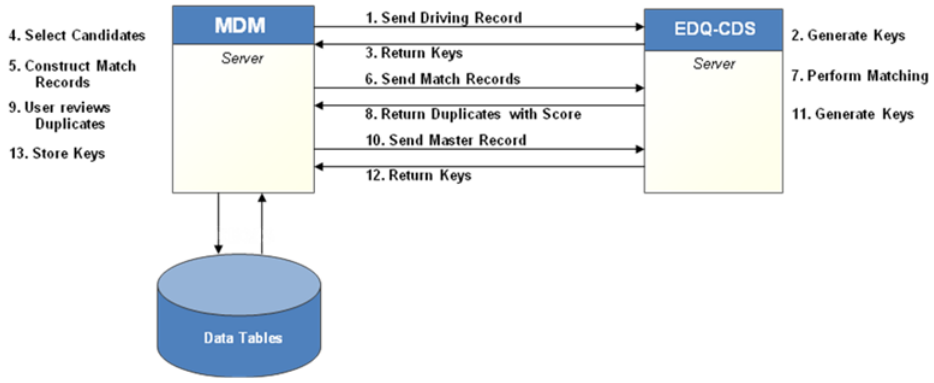
Attribute Name	Data Type	Notes
`serverid`	String	Server ID. Not applicable to Siebel.
`jobid`	String	Job ID. Not applicable to Siebel.
`externalid`	String	ID of the individual, entity, or address of the key generated record.
`keyprofile`	String	Key profile of the generated key.
`keyvalue`	String	Key value.
`keypriority`	Number	Key priority

5.5 Real-Time Integration

The EDQ-CDS real-time matching services can be called by an external application without any changes to the default configuration. It is the responsibility of the calling application to manage the storage of record keys and to perform the selection of match candidates to be passed to the matching service.

A typical interaction between the calling application (for example, a CRM or Master Data Management [MDM] application) and EDQ-CDS during real-time matching (for example, Contact duplicate prevention) is illustrated as follows:

Figure 5–1 Overview of Expected Integration Architecture with Matching Services



In detail the matching services operate and are used as follows:

- **Send Driving Record** — The application sends the new (driving) record and the configured key profile to EDQ-CDS.
- **Generate Keys** — EDQ-CDS generates the key(s) for the driving record.
- **Return Keys** — EDQ-CDS returns the driving record's keys to the MDM application.
- **Select Candidates** — The MDM application selects all (candidate) records that share any of the same generated keys. If no candidates are identified then go to the Store Keys.
- **Construct Match Data** — The MDM application constructs the match data for the driving and candidate records
- **Send Match Records** — The MDM application sends the data for the new (driving) record and candidates to EDQ-CDS.
- **Perform Matching** — EDQ-CDS matches the driving record against the candidates to identify potential duplicates. Each match is assigned a score indicating the strength of match.
- **Return Duplicates with score** — EDQ-CDS returns the IDs of the matched candidates (and scores) to the MDM application. The driving record is also returned, but with a blank score. If no duplicates were identified by EDQ-CDS then go to the Store Keys.
- **User reviews Duplicates** — As indicated.
- **Send Master Record** — If duplicates were identified by EDQ-CDS and selected by the user, then the driving record is merged with the existing duplicate record. If a merge operation occurred then the MDM application sends the new merged (master) record details back to EDQ-CDS.
- **Generate Keys** — EDQ-CDS uses the details of the master record to generate key values.
- **Return Keys** — EDQ-CDS returns the master record's keys to the MDM application.
- **Store Keys** — The MDM application stores the keys for the new master record.