Oracle® Fusion Middleware

Integrating and Configuring Siebel Environments with Enterprise Data Quality

12*c* (12.2.1.4.0)
E95648-01
September 2019

ORACLE®

Oracle Fusion Middleware Integrating and Configuring Siebel Environments with Enterprise Data Quality, 12*c* (12.2.1.4.0)

E95648-01

Contents

Preface

1 Planning the Enterprise Data Quality Siebel Connector Integration

2 Integrating the Enterprise Data Quality Siebel Connector

3 Configuring the Enterprise Data Quality Siebel Connector

ORACLE

Preface

This document describes how to integrate and configure the Oracle Enterprise Data Quality Customer Data Services Pack Siebel Connector with a Siebel server.

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 readers are familiar with Web technologies and have a general understanding of Windows and UNIX 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 documentation set.

EDQ Documentation Library

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

- *Release Notes for Enterprise Data Quality*
- *Installing and Configuring Enterprise Data Quality*
- *Administering Enterprise Data Quality*
- *Understanding Enterprise Data Quality*
- *Integrating Enterprise Data Quality With External Systems*
- *Securing Oracle Enterprise Data Quality*
- *Installation and Upgrade Guide*
- *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 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.

1

Planning the Enterprise Data Quality Siebel Connector Integration

This chapter helps to prepare you for your Enterprise Data Quality (EDQ) Siebel Connector installation and integration to an existing Siebel server.
This chapter includes the following sections:

- Understanding the Enterprise Data Quality Siebel Connector
- Verifying the Siebel Server and Instance
- Preparing for the Enterprise Data Quality Siebel Connector Integration

The EDQ Siebel Connector is a small footprint application that resides on the Siebel server and enables communication with EDQ for real-time (as records are inserted and updated into Siebel) and batch (for running regular data quality jobs on data in Siebel) data quality services. To enable real-time services, the EDQ Siebel Connector translates Siebel function calls into web service requests for EDQ. For batch jobs, EDQ jobs are called from the Siebel Data Quality Manager and a shared staging database is used to pass data between Siebel and EDQ.

Understanding the Enterprise Data Quality Siebel Connector

The EDQ Siebel Connector calls EDQ jobs (for batch matching and health checks) and web services (for real-time cleansing and matching). A shared staging database is used to pass data between Siebel and EDQ-CDS when running batch jobs.

Understanding the EDQ Siebel Connector Architecture

The high-level architecture of the EDQ Siebel Connector is illustrated in the following diagram (assuming a single EDQ server):

Figure 1-1   EDQ Siebel Connector Architecture
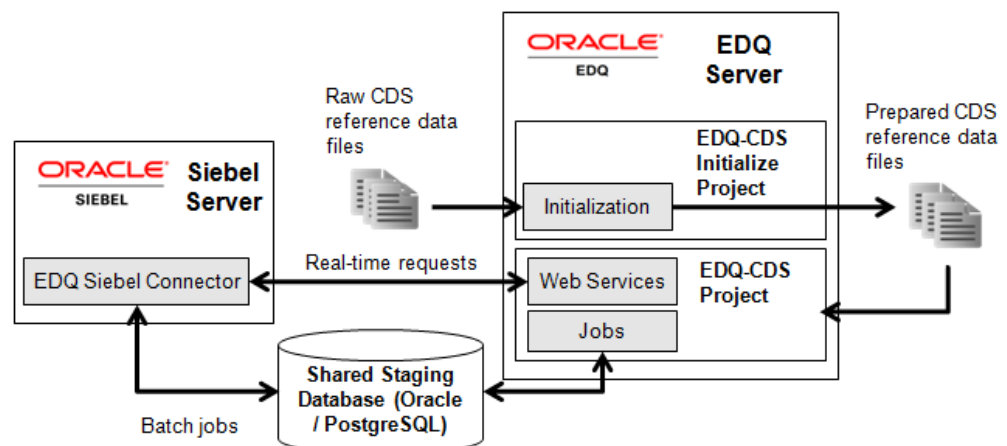
Prior to installation, it is important to become familiar with Siebel's universal data quality interface. For more information, see *Siebel Data Quality Administration Guide*

Using EDQ-CDS With Siebel CRM

EDQ can be deployed to protect and monitor the quality of data in a Siebel Customer Relationship Management (CRM) environment. When attached to Siebel CRM, EDQ can:

- Prevent duplicate contacts, account and prospect records from being added to the system by automatically matching all new and updated records against the other records in the system

- Standardize account, contact, prospect, and address data as it is entered or updated in the system

- Perform batch duplicate identification and cleansing tasks

- Measure the quality of account, contact, prospect, and address data on both an ad-hoc, and a regular basis

As Siebel CRM has only a single set of data to manage, EDQ deployment is simple. When records are added or updated in Siebel CRM, the EDQ real-time interface is used to standardize the data, and then match it against existing data.

The EDQ real-time interface is also used for batch cleansing (standardization) tasks: Siebel sends each record selected for batch processing to the EDQ cleansing web services.

Batch duplicate identification and data quality health check jobs (either Full Batch or Incremental Batch) use the EDQ batch interface. In this case, the driver and candidate records for matching or health checks are written to the shared staging database. EDQ then runs a batch job on these records and when matching occurs the matches are written back to the shared staging database, which Siebel picks up and uses to link records together.

Using EDQ-CDS With Siebel UCM

When deployed with Siebel Universal Customer Master (UCM), EDQ provides the essential matching engine for the hub. As records are added or updated in source systems they are automatically matched against the master data in Siebel UCM in order to maintain the quality of the master data and its linkage with the source data records.

The EDQ services provided to Siebel UCM are the same as those in Siebel CRM though the additional UCM capabilities are used to manage record survivorship and best record generation. Siebel UCM also supports automatic matching when EDQ returns a match score over a specified threshold.

> **Note:**
>
> The UCM Batch workflow invokes the EDQ real-time interface for standardization and matching, so that a number of records can be loaded into the hub and automatically matched and reconciled with the existing master data. The EDQ batch interface can still be invoked from Siebel UCM, but works only with the master data records.

Migrating High Volumes of Data

When migrating a high volume of data records into Siebel UCM (i.e over 10m records), it is best practice to match and merge the records in EDQ before loading them into Siebel (normally via Siebel's EIM interface), as opposed to matching them using the UCM Batch workflow and the real-time interface. This ensures considerably greater efficiency in the matching process as all in-scope records are matched against each other and all matches found in a single operation, rather than continuously matching each record against the records that have been loaded so far into the hub.

Verifying the Siebel Server and Instance

Before installing the EDQ Siebel Connector, you must ensure that the Siebel server and the instance residing on it conform to the following criteria:

- EDQ-CDS 12*c* (12.2.1.1) requires the 12*c* release of the EDQ Siebel Connector (included with EDQ 12.2.1).

- Siebel CRM or UCM version 8.1 or later, with a Siebel Data Quality license.

- The Java Runtime Environment (JRE) version 7 or later must be installed on the Siebel server.

- The version of JRE deployed must match the architecture of the server supported by Siebel. For example, Siebel only supports 32-bit versions of Windows so the 32-bit version of JRE must be installed.

Preparing for the Enterprise Data Quality Siebel Connector Integration

When integrating a Siebel instance with EDQ Customer Data Services Pack (EDQ-CDS), Oracle recommends that the necessary components be installed and configured in the following order:

1. Install the EDQ-CDS pack on the EDQ server. For more information, see *Customer Data Services Pack Guide for Enterprise Data Quality*.

2. Install the EDQ Siebel Connector on the Siebel server, see.Integrating the Enterprise Data Quality Siebel Connector .

3. Integrate Siebel with EDQ-CDS as detailed in Configuring the Enterprise Data Quality Siebel Connector ..

The EDQ Siebel Connector is shipped with EDQ in the EDQ 12*c* (12.2.1.1) Media Pack. The installation instructions vary depending on the platform of the Siebel server.

2

Integrating the Enterprise Data Quality Siebel Connector

This chapter describes how to integrate the Enterprise Data Quality (EDQ) Siebel Connector with an existing Siebel server.
This chapter includes the following sections:

- Integrating the EDQ Siebel Connector With a Windows Siebel Server

- Integrating the Connector With AIX, Linux, and Solaris Siebel Servers

- Resolving Conflicts Between Siebel Java Business Services and the EDQ Siebel Connector

Integrating the EDQ Siebel Connector With a Windows Siebel Server

Use the following steps to integrate the connector with a Windows Siebel server. The `siebelconnector.zip` is automatically installed into the EDQ Oracle Home when you run the EDQ installer.

1. Copy the `siebelconnector.zip` file from the EDQ Oracle Home to the installation directory on the Siebel server.

2. Extract the `siebelconnector.zip` file into the installation directory specified in the `dnd.parms` file (the default being `Siebel_Server_root\dnd\install` .)

    This installs the following files:

    - Jar files for the connector.

    - Library jar files needed by the connector.

3. Copy the `dnd.dll` file to the `Siebel_Server_root\bin\` directory.

4. Create a file in the `Siebel_Server_root\SDQConnector\` directory with the name `dnd.parms`. This file will be used to point to the installation directory on the same server where the Siebel connector will run. The file must contain the following lines:

    ```
    javalib: [location of the JRE]/bin/client/jvm.dll
    directory: [the installation directory for the connector]
    ```

    For example:

    ```
    javalib: C:/Program Files/Java/jre1.7/bin/client/jvm.dll
    directory: C:/SiebelConnector
    ```

ORACLE

> **✏ Note:**
>
> The Siebel connector is not language specific so is not installed in a specific language directory.

5. (Optional) If detailed trace information on connector requests is required (for example, for temporary debugging purposes) add the following lines to the `dnd.parms` file to enable logging:

```
logfile: sdq%05d.log
jlogfile: jsdq%05d.log
```

6. The `dnd.properties` file in the Siebel Connector installation directory configures how the connector communicates with EDQ. In most cases, the connector is configured to connect to an EDQ server with EDQ-CDS installed. Some settings in this file will need editing to reflect your particular environment; for more information, see Configuring the EDQ Siebel Connector.

Integrating the Connector With AIX, Linux, and Solaris Siebel Servers

Use the following steps to integrate the connector with an AIX, Linux, or Solaris Siebel server. The `siebelconnector.zip` is automatically installed into the EDQ Oracle Home when you run the EDQ installer.

1. Copy the `siebelconnector.zip` file from the EDQ Oracle Home to the installation directory on the Siebel server.

2. Extract the `siebelconnector.zip` file into the installation directory specified in the `dnd.parms` file (the default being `/opt/siebel/dnd/install`).

   This installs the following files:

   • Jar files for the connector

   • Library jar files needed by the connector

3. Copy the `libdnd.so` driver file to the `Siebel_Server_root/lib directory`. The driver files for each OS are held in the `native` sub-directory of the `siebelconnector.zip` file in the EDQ distribution, as follows:

   AIX:
   `/native/aix/ppc/` or `/native/ppc64`, depending on whether a 32-bit or 64-bit version of Siebel is in use.

   Linux:
   `/native/linux`

   Solaris:
   `/native/sparcv9`

4. Create a file in the `Siebel_Server_root/SDQConnector` directory with the name `dnd.parms`. This file is used to point to the installation directory on the same server where the EDQ Siebel Connector will run. The file must contain the following lines:

```
javalib: [location of the JRE]/lib/i386/client/libjvm.so
directory: [the installation directory for the connector]
```

For example:

```
javalib: /usr/java/jre1.7/lib/i386/client/libjvm.so
directory: /opt/siebel/dnd/install
```

> **Note:**
>
> The Siebel Connector requires the Oracle JRE/JDK version 1.7 or later.

> **Note:**
>
> Java 8 is not available for Solaris 32-bit, the connector must be run in remote mode so that it can run using a 64-bit JRE.

5. If detailed trace information on connector requests is required, for example for temporary debugging purposes, add the following lines to the `dnd.parms` file to enable logging:

```
logfile: sdq%05d.log
jlogfile: jsdq%05d.log
```

6. The `dnd.properties` file in the Siebel Connector installation directory configures how the connector communicates with EDQ. In most cases, the connector is configured to connect to an EDQ server with EDQ-CDS installed. Some settings in this file will need editing to reflect your particular environment; for more information, see Configuring the EDQ Siebel Connector.

Resolving Conflicts Between Siebel Java Business Services and the EDQ Siebel Connector

In scenarios where Siebel Java Business Services (like the Siebel Java Message Service) are deployed, the EDQ Siebel Connector may conflict with them causing it to fail.

To avoid this conflict, the connector must be configured post-installation to connect to a separate Java process using the following steps:

1. Open the `dnd.parms` file, and add the following line to turn on remote mode:

```
remote: true
```

The `directory` and `javalib` lines are no longer used, although they can be left in the file.

2. In the Siebel Connector installation directory, run the following command to start the Java code as a server:

```
java -jar connector.jar [-p portnum] [directory_path]
```

Where:

- `-p` *portnum* should be used only if the connector must listen on a port other than the default port of 8642. Specify an available TCP/IP port number to use. If using this argument, also add the same port number to the `dnd.params` file, as follows:

  `port:` *portnum*

- *directory_path* should be used only if the `dnd.properties` file is located somewhere other than in the same directory as the `connector.jar` file (the default). Specify the path to the directory where `dnd.properties` is stored.

> **Note:**
>
> The version of the JRE can be version 1.7 or later, 32- or 64-bit. The JRE must be either the Oracle (Sun) JRE, or the IBM JRE.

3. Restart Siebel and perform further tests.

In some instances, the `install` directory (containing the `.jar` files and `dnd.properties`) is not on the same system as the Siebel instance. If this is the case, add a `host` line to `dnd.parms` to specify the host running the Siebel Java code (for example, `host:` *HOSTNAME*).

3

Configuring the Enterprise Data Quality Siebel Connector

This chapter describes how to configure the EDQ Siebel Connector with an existing Siebel server.

This chapter includes the following sections:

Configuring the EDQ Siebel Connector

The EDQ Siebel Connector requires a configuration file called `dnd.properties`, which must be installed in the location specified in the `dnd.parms` file (installed and configured as part of the EDQ Siebel Connector.) The settings in the `dnd.properties` file control which EDQ server and project are used to provide data quality services to a Siebel instance, and how to connect to it.

Understanding the Settings

The EDQ Siebel Connector settings are in the `dnd.properties` file and fall into the following categories:

- CDS Connection Settings

- Multiple Child Entity groupings

- Real-Time Service Definitions

- Batch Job Definitions

CDS Connection Settings

These settings are used to connect to an server for running jobs and for real-time services. When setting these, make sure there are no trailing spaces in the values as these will cause errors when attempting to establish a connection.

- `httpprefix.cds` - the hostname, HTTP port and context name of the server (for example `http://hostname:port/edq/webservices`).

> **Note:**
>
> Although requires SSL to be used for communication between its own web pages and the server, and therefore redirects any http requests to its web pages to the https port, http can still be used for web service requests if required (and allowed by security standards). If is installed using the Windows installer, the default http port is 9002. On other installations, the port is specified when is deployed onto the application server. If the use of https is a requirement, the https prefix and port number (which defaults to 9004 if is installed using the Windows installer), can be specified here, but in order to establish a trusted connection with , it is also necessary to import either the certificate, or the certificate's root, from the application server into the Certificate Store of the JRE used by the Siebel Connector (using the standard Java `keytool` command).

• `jmxserver.cds` - the hostname and port of the server's JMX interface (for example, `servername:9005`).

> **Note:**
>
> If is installed using the Windows installer, the JMX management port defaults to 9005. On other installations, it defaults to 8090. It can be checked by reading the `management.port` entry in the file `director.properties` in the config directory of the server. If this entry is not present, the port will be 8090.

• `username.cds` and `password.cds` - The login credentials for a designated JMX and web services user on the server. This account must have the following permissions (at minimum):

  – **System: Connect to Messaging System**, which grants access to submit web service requests or JMS messages to .

  – **Server Console: Execute Job** and **Director: Execute Job**, which allows you to execute jobs

  – Permission to the project stated under `projectname` to verify that the user has project permission, right-click on the project in 's Director application, select **Properties**, then select the **Security** tab. Check that the user is a member of a group with access to the project. If the user account previously described was the account used to import the project from a DXI file, it must have permission.

• **Staging Database connection details** - The connection details and credentials of the Staging Database used to pass data for batch matching jobs between Siebel and -CDS. For further information, see Configuring the Staging Database.

• `projectname.cds` is the name of the -CDS project on the server. This setting can be left unchanged if the project has not been renamed.

• `projectname.cdshc` is the name of the -CDS Health Check project on the server. This setting can be left unchanged if the project has not been renamed.

Multiple Child Entity Groupings

Hierarchical data in Siebel is transformed by the Siebel Connector into a flat record structure so that can match 'parent' records (such as Contacts, Prospects and Accounts) using the details of multiple 'child' records (such as Names, Addresses, Email Addresses, Alternative Phone Numbers etc.)

The Multiple Child Entity settings specify how scalar (single field) and non-scalar (multi-field) child entities are handled by the Siebel Connector.

> **Note:**
>
> These settings are only required if Multiple Child Entities are enabled in - CDS and supported by the installed version of Siebel (version 8.1.1.10 or later), see Understanding the Vendor Parameters and the Siebel documentation for further details.

Scalar Groups

Scalar groups are used for Siebel Business Components comprising one field, or for which only one field is mapped in the Data Quality field mappings in Siebel. The connector prepares the values for such groups into a simple delimited list of values in a single input attribute for .

The required configuration format is:

`group.[name] = [BC1],[BC2],...`

`group.[name].concat = [EDQ-CDS Attribute Name]`

`group.[name].delimiter = [Delimiter character]`

where:

- `name`: label applied to the business components.
- `BCn`: Siebel Business Component name.
- `group.[name].concat`: the attribute under which the data will be concatenated.
- `group.[name].delimiter`: the delimiter character used to separate the data.

For example, the following group in the default `dnd.properties` file sets this configuration for the `AlternatePhone` business component in Siebel:

```
group.altphone = AlternatePhone
group.altphone.concat = alternatephone
group.altphone.delimiter = |
```

So all `AlternatePhone` values for each parent entity are concatenated into a delimited list.

This means that `AlternatePhone` values for Contacts in Siebel that are stored in this format:

```
contact1
    AlternatePhone1
```

```
        AlternatePhone2
contact2
        AlternatePhone3
        AlternatePhone4
```

are transformed into this format:

```
contact1 AlternatePhone1|AlternatePhone2
contact2 AlternatePhone3|AlternatePhone4
```

Non-Scalar Groups

Non-scalar groups are used for more complex child entities comprising multiple fields, such as names and addresses. In this case, the connector prepares multiple records for the same parent to submit to for matching purposes, each with different child data.

The required configuration format is:

```
group.[name] = [BC1],[BC2]...
```

Be sure to replace the following with your information:

- *name*: label applied to the business components

- *BCn*: Siebel Business Component name

For example, the following non-scalar group specifies a list of Siebel Business Components, each storing addresses:

```
group.address = CUTAddress,PersonalAddress
```

This setting means that CUTAddress and PersonalAddress records are grouped under the address label by the Connector and expanded into separate records for matching in -CDS.

The CUTAddress and PersonalAddress values for Contacts format is:

```
contact1
        CUTAddress1
        PersonalAddress1
contact2
        CUTAddress2
        PersonalAddress2
```

The values are transformed into this format:

```
contact1 address1 [CUTAddress1]
contact1 address2 [PersonalAddress1]
contact2 address3 [CUTAddress2]
contact2 address4 [PersonalAddress2]
```

Real-Time Service Definitions

These properties are used to configure how the Siebel Connector communicates with -CDS over web services.

> **Note:**
>
> It is not normally necessary to change these properties, as the Global Communication settings are inherited.

The `dnd.properties` file comes with 10 pre-defined web services configurations, `ws1` to `ws10`. These are pre-configured with some or all of the following parameters, which can be edited if required:

- `type` - A tag identifying the Data Quality operation in the Siebel vendor parameters.
- `conduit` - Specifies how the Siebel Connector communicates with -CDS. Possible values are `simplews`, `dbbatch`, `jmsbatch`.
- `url` - The URL end point of the web service.
- `failsafe` - Specifies how the Connector behaves in the event of an error:
    - `true`: the Connector fails silently, for example, no error message is generated.
    - `false`: the Connector generates an error message.
- `username` and `password` - The credentials used to connect to the web service.
- `parameternames` - A comma-separated list of Siebel session parameters to be passed to -CDS as web service parameters.
- `idelement` - the name of the XML element generated in the response to Siebel.

Batch Job Definitions

The Batch Job Definitions are divided into 8 groups as follows:

- Account Match
- Account Health Check
- Account Key Analysis
- Contact Match
- Contact Health Check
- Contact Key Analysis
- Prospect Match
- Prospect Key Analysis

These groups all have properties that control the way Batch operations are run.

These properties control how data flows between the Siebel Connector and -CDS:

- `type` - A tag identifying the Data Quality operation. This must match the in Siebel vendor parameters.
- `conduit` - Specifies how the Siebel Connector communicates with -CDS. Possible values are `simplews`, `dbbatch`, `jmsbatch`.
- `failsafe` - Specifies how the Connector behaves in the event of an error:
    - true: the Connector fails silently, for example, no error message is generated.

ORACLE

- false: the Connector generates an error message.

- `writetable` - the Candidates table in the staging database that the Connector writes data to, and that -CDS reads data from.

- `db` - The database connection string for the staging database.

- `project` - CDS project name, which reads from the `$projectname` variable at the start of the dnd.properties file.

- `job` - The -CDS job name.

- `runlabel` - The Run Label for the job.

- `overrides` - The values of these settings and the Staging Database override the default (externalized) configuration settings in the -CDS processes. For example, Key Profile, Match Threshold (`Typical` and `70` respectively, by default.) However, they can in turn be overridden by parameters configured in Siebel, see Understanding the Vendor Parameters.

- `readtable` - Name of the table in the staging database that matches are written to.

- `idelement` - The identifier element for the Siebel Business Component.

Configuring Siebel to Use Customer Data Services

The following is a step-by-step guide to configuring Siebel to use EDQ-CDS. Please read the Siebel Data Quality Administration Guide (available as part of Siebel Bookshelf) before attempting to follow these steps.

> **✎ Note:**
>
> Siebel 8.1.1.10 or later is configured by default with all the settings described in this chapter. Therefore, if using this version of Siebel, this chapter can be used as a check list to confirm all the settings are correct. For older versions of Siebel, please use the settings described in this guide except where stated.

The guide is in four sections:

- Server Configuration
- Data Quality Administration
- User Preferences
- (Optional) Creating Templates to Enable Batch Data Quality Jobs

Server Configuration

1. Log into the Siebel web client as a system administrator.
2. Navigate to **Administration - Server Configuration** on the Siebel Site Map.
3. Click **Enterprises** at the top of the page and select the **Component Groups** tab.
4. Find the **Data Quality** component group and ensure it is enabled.

5. Check that the component group has been assigned and enabled on the Siebel server, `cai`, in the following example:

6. Switch to the **Parameters** tab and set the **Data Cleansing Type** parameter to **EDQ**:

7. Set the **DeDuplication Data Type** parameter to **EDQ**:

8. Click **Servers** at the top of the page, find the **Data Quality Manager** component and select the **Parameters** tab.

9. Ensure the following parameter values are set to **EDQ**:

   • Data Cleansing Type

   • DeDuplication Data Type

10. Set the following parameter values to **True**:

   • Data Cleansing Enable Flag

   • DeDuplication Enable Flag

Note:

The preceding illustration shows the values after a server restart, which is performed when Siebel configuration is completed.

11. Repeat Step 8 to Step 10 for the Siebel application components you want to use with EDQ-CDS. For example:

- Call Center Object Manager

- EAI Object Manager

- Sales Object Manager

Tip:

You can use these steps for any other components that you want to use with EDQ-CDS.

Data Quality Administration

To set the Data Quality Administration options, use the following procedure:

1. Log into Siebel as a system administrator.

2. Navigate to **Administration - Data Quality** on the Siebel Site Map.

3. Click **Data Quality Settings** and create the following options:

   • **Enable DataCleansing = Yes**

   • **Enable DeDuplication = Yes**

   Add the following options to enable a pop-up dialog of possible duplicate records for each new record of a given type when the real-time duplicate prevention service is running. If these options are not specified, and the real-time duplicate prevention service is running, the check will still be performed, but you must actively navigate to the **Duplicate Accounts**, **Duplicate Contacts** or **Duplicate Prospects** tab when adding a new record:

   • **Force User DeDupe - Account = Yes**

   • **Force User DeDupe - Contact = Yes**

   • **Force User DeDupe - List Mgmt = Yes**

4. Click **Third Party Administration**. Add a new Vendor entry with a Name of **EDQ**, and a DLL Name of **dnd**:
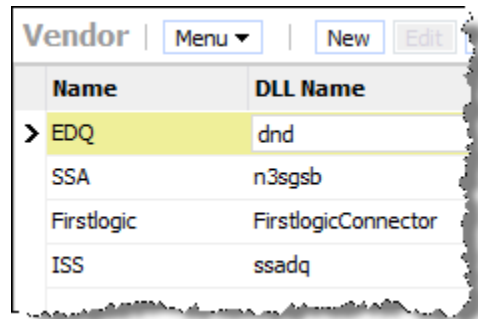
5. Select the EDQ vendor selected and add **BC Operations** as follows:

Business Component Name	Operations
Account	Data Cleansing
Account	DeDuplication
Contact	Data Cleansing
Contact	DeDuplication
CUT Address	Data Cleansing
List Mgmt Prospective Contact	Data Cleansing
List Mgmt Prospective Contact	DeDuplication

Additionally, add the following Business Components to configure matching using multiple child addresses:

Business Component Name	Operations
CUT Address	DeDuplication
Personal Address	Data Cleansing

Business Component Name	Operations
Personal Address	DeDuplication

6. Scroll to the bottom of the screen and set the **Field Mappings** for each BC Operation. These mappings are detailed in Understanding the Field Mappings for Business Components.

> ✎ **Note:**
>
> If you add or change field mappings, then they must correspond to the configured web services in EDQ (for real-time operations) and the names of the columns in the tables of the staging database (for batch operations). Also, you *must* modify the relevant DQ Integration Objects and Business Services using Siebel Tools. If you are using Siebel UCM, then the you must modify the data maps between UCM Integration Objects and DQ Integration Objects.

7. With the EDQ vendor selected, select the **Vendor Parameter** tab in the middle section of the screen and add the parameters specified in Understanding the Vendor Parameters.

User Preferences

To enable data quality for any user, use the following procedure:

1. Select **User Preferences** on the **Tools** menu.

2. Click **Data Quality**.

3. Set both the Enable DeDuplication and Enable Data Cleansing options to **Yes**.

This completes the Siebel configuration. Restart the Siebel server to ensure that all the configuration changes have taken effect.

(Optional) Creating Templates to Enable Batch Data Quality Jobs

Siebel can be configured to run batch data quality jobs from the Server Management UI. Jobs can also be run from the Siebel command line, or job configurations can be stored in files and reused as required.

To do this, some custom Job Templates must be added to Siebel. These templates and the parameters required are listed in Understanding the Job Template Configuration.

To add a Job Template:

1. Open the Siebel web client.

2. Navigate to **Administration - Server Configuration** on the Siebel Site Map.

3. Click **Job Templates**.

4. Click **New** to create a new Job Template.

5. Complete the fields in the Job Templates and Job Parameters area using the details provided in Understanding the Job Template Configuration.

> ✎ **Note:**
>
> Set the Component field to Data Quality Manager for each new Job
> Template. If there are no options listed in this drop-down field, navigate
> to the **Component Definitions** tab on the **Enterprise** screen, and click
> Synchronize.

Create a new Job Template for every job listed in Understanding the Job Template
Configuration.

Configuring the Staging Database

The Staging Database is used by the Siebel Connector as a staging area for handing
over data between Siebel and EDQ when running batch jobs through Siebel's Data
Quality Manager.

This batch interface is most commonly used when EDQ is connected to a standalone
Siebel CRM system. In Siebel UCM, the 'UCM Batch' flow does not use this interface
when matching inbound data against the master data in Siebel UCM. Rather, it calls
out to EDQ's real-time matching services.

However, for both CRM and UCM the Siebel Data Quality Manager service can run
batch duplicate identification and health check jobs on the master data only.

When such batch jobs are run, driver and candidate records for matching and input
records for the Data Quality Health Check service are written to tables in the staging
database, to be read by an EDQ job. For the matching service only, EDQ then exports
duplicates from the matching process to another table in this staging database to be
read back into Siebel.

Creating Tables

The tables are created by the RCU during the installation of EDQ.

Configuring Connections

Both the Siebel Connector and EDQ-CDS itself need to connect to the Staging
Database in order to read and write to the Candidate and Match tables when
processing Batch jobs. These tables can be created in any schema of a supported
database type (Oracle). The default connection string is for an Oracle database.

The connection details are specified in the `dnd.properties` file, see Configuring the
EDQ Siebel Connector. To configure these, open the `dnd.properties` file and edit the
parameters in the **CDS Connection Settings** section near the top of the file. These
parameters control the database host, port, data source JNDI name, credentials and
other settings used to connect to the Staging Database.

Finalizing and Verifying the Configuration

The following sections describe how to finalize the system and verify that the EDQ
Siebel Connector is installed and configured correctly.

Before starting, ensure that the appropriate real-time jobs are running in EDQ, see *Oracle Fusion Middleware Installing and Customizing Enterprise Data Quality Customer Services Data Services Pack*.

Generating Cluster Keys

If the Siebel database is already populated with records (`Accounts/Contacts/Prospects`) then the cluster keys in the system must be generated or refreshed for the new configuration before it is used.

In Siebel, run the following jobs (as defined in Understanding the Job Template Configuration) from the **Administration > Server** page:

- Generate account keys
- Generate contact keys
- Generate prospect keys

> **Note:**
>
> If is being used for key generation, these jobs will automatically call key generation web services instead of using the configured Token Expression in Siebel to generate the keys.

Testing the Account Cleaning Service

To test the cleaning service for Account records:

1. Check that data cleansing is enabled for the server by ensuring that **Administration - Data Quality > Data Quality Settings > Enable Data Cleansing** is set to "Yes".

   > **Note:**
   >
   > If this setting is changed from `No` to `Yes`, it is necessary to restart the Siebel Server.

2. Check that data cleansing is enabled for the current user profile by ensuring that **Tools > User Preferences > Data Quality > Enable Data Cleansing** is set to "Yes".

   > **Note:**
   >
   > If this setting is changed from `No` to `Yes`, it is necessary to exit the current client session and log in again as the same user.

3. Enter a new Account record with an Account Name in lower case letters.

4. Save the record.

5. Check that the Account Name has been converted to upper case to verify that the cleaning service has been called.

Testing the Batch Matching Service

To test the batch matching service for Contact records:

1. Temporarily disable real-time deduplication for the current user profile by setting **Tools > User Preferences > Data Quality > Enable DeDuplication** to **No**.

> **Note:**
>
> If this setting is changed from `No` to `Yes` it is necessary to restart the Siebel Server.

2. Enter and save two new Contact records with the same name and email address.

3. Run the Batch contact match job from the **Administration - Server** page

4. Verify that the records were matched by checking the **Administration - Data Quality > Duplicate Contacts** page.

Testing Real-Time Contact Matching

To test the real-time matching service for Contact records:

1. Check that real-time deduplication is enabled for the current user profile by checking that **Tools > User Preferences > Data Quality > Enable DeDuplication** is set to **Yes**.

2. Set **Administration - Data Quality > Data Quality Settings > Force User DeDupe - Contact** to **Yes**.

3. Enter a new Contact record with the same details as in Testing the Batch Matching Service.

4. Save the record.

5. Verify that the **Possible Matching Contacts** dialog is displayed showing matches against the two contact records created previously.

> **Note:**
>
> An exact name and email match, as in this example, will only match if is being used for key generation (for Siebel 8.1.1.10 or later), or if the Query and Token expressions used for key generation have been adjusted to use only name, or name and email, attributes.

Understanding the Job Template Configuration

For clarity, the `Data Quality Setting` parameter found in most of the Templates listed in the table begins with a single double quote mark (") and two single (') quote marks, and ends with one single quote mark and one double.

Name	Short Name	Parameters
Batch account cleanse	BatAccClean	Buscomp Name = Account
		Business Object Name = Account
		Operation Type = Data Cleansing
Batch account match	BatAccMatch	Buscomp Name = Account
		Business Object Name = Account
		Data Quality Setting = "",'Yes','dedupe'"
		Operation Type = DeDuplication
Batch account health check	BatAccHealth	Buscomp Name = Account
		Business Object Name = Account
		Data Quality Setting = "",'Yes','healthcheck'"
		Operation Type = DeDuplication
Batch account key analysis	BatAccKeyAnalysis	Buscomp Name = Account
		Business Object Name = Account
		Data Quality Setting = "",'Yes','keyanalysis'"
		Operation Type = DeDuplication
Batch address cleanse	BatAddClean	Buscomp Name = CUT Address
		Business Object Name = CUT Address
		Operation Type = Data Cleansing
Batch contact cleanse	BatConClean	Buscomp Name = Contact
		Business Object Name = Contact
		Operation Type = Data Cleansing
Batch contact match	BatConMatch	Buscomp Name = Contact
		Business Object Name = Contact
		Data Quality Setting = "",'Yes','dedupe'"
		Operation Type = DeDuplication
Batch contact health check	BatConHealth	Buscomp Name = Contact
		Business Object Name = Contact
		Data Quality Setting = "",'Yes','healthcheck'"
		Operation Type = DeDuplication
Batch contact key analysis	BatConKeyAnalysis	Buscomp Name = Contact
		Business Object Name = Contact
		Data Quality Setting = "",'Yes','keyanalysis'"
		Operation Type = DeDuplication
Batch prospect cleanse	BatProClean	Buscomp Name = List Mgmt Prospective Contact
		Business Object Name = List Mgmt Prospective Contact
		Operation Type = Data Cleansing

Name	Short Name	Parameters
Batch prospect match	BatProMatch	Buscomp Name = List Mgmt Prospective Contact
		Business Object Name =List Mgmt Prospective Contact
		Data Quality Setting = "",'Yes','dedupe"'
		Operation Type = DeDuplication
Batch prospect key analysis	BatProKeyAnalysis	Buscomp Name = List Mgmt Prospective Contact
		Business Object Name =List Mgmt Prospective Contact
		Data Quality Setting = "",'Yes','keyanalysis"'
		Operation Type = DeDuplication
Generate account keys	GenAccKey	Buscomp Name = Account
		Business Object Name = Account
		Operation Type = Key Generate
Generate contact keys	GenConKey	Buscomp Name = Contact
		Business Object Name = Contact
		Operation Type = Key Generate
Generate prospect keys	GenProKey	Buscomp Name = List Mgmt Prospective Contact
		Business Object Name = List Mgmt Prospective Contact
		Operation Type = Key Generate
Incremental account match	IncAccMatch	Buscomp Name = Account B
		usiness Object Name = Account
		Data Quality Setting = "",'No','dedupe"'
		Operation Type = DeDuplication Object
		Where Clause = [Updated] > '12/18/2007 00:00:00'
		Note: The preceding Object Where Clause is an example only.
Incremental account health check	IncAccHealth	Buscomp Name = Account
		Business Object Name = Account
		Data Quality Setting = "",'No','healthcheck"' Operation Type = DeDuplication Object
		Where Clause = [Updated] > '12/18/2007 00:00:00'
Incremental contact match	IncConMatch	Buscomp Name = Contact
		Business Object Name = Contact
		Data Quality Setting = "",'No','dedupe"'
		Operation Type = DeDuplication Object
		Where Clause = [Updated] > '12/18/2007 00:00:00'
		Note: The preceding Object Where Clause is an example only

Name	Short Name	Parameters
Incremental contact health check	IncConHealth	Buscomp Name = Contact
		Business Object Name = Contact
		Data Quality Setting = "",'No','healthcheck'"
		Operation Type = DeDuplication Object
		Where Clause = [Updated] > '12/18/2007 00:00:00'
Incremental prospect match	IncProMatch	Buscomp Name = List Mgmt Prospective Contact
		Business Object Name = List Mgmt Prospective Contact
		Data Quality Setting = "",'No','dedupe'"
		Operation Type = DeDuplication Object
		Where Clause = [Updated] > '12/18/2007 00:00:00'
		Note: The preceding Object Where Clause is an example only.
Refresh account keys	RefAccKey	Buscomp Name = Account
		Business Object Name = Account
		Operation Type = Key Refresh
Refresh contact keys	RefConKey	Buscomp Name = Contact
		Business Object Name = Contact
		Operation Type = Key Refresh
Refresh prospect keys	RefProKey	Buscomp Name = List Mgmt Prospective Contact
		Business Object Name = List Mgmt Prospective Contact
		Operation Type = Key Refresh

Understanding the Field Mappings for Business Components

> **Note:**
>
> All fields in these Business Components are drop-down fields, with the exception of `Id`. This field must be completed manually.

UIDs, EIDs, and IEIDs

UIDs, EIDs, and IEIDs are new optional EDQ mapped fields which can mapped to any Siebel business component field as required by the user.

The Mapped Fields are:

- UID - uid1, uid2, uid3

- EID - eid1, eid2, eid3
- IEID - ieid1, ieid2, ieid3

The Mapped Fields apply to the following operations:

- Account - DeDuplication
- Contact - DeDuplication
- List Mgmt Prospective Contact - DeDuplication

Account - Data Cleansing

Business Component Field	Mapped Field
Id	entityid
Language Code	languages
Name	name
Location	subname
Main Phone Number	phone
Tax ID Number	taxnumber
Home Page	website
VAT registration number	vatnumber

Account - DeDuplication

Business Component Field	Mapped Field
Id	entityid
Language Code	languages
Name	name
Location	subname
Main Phone Number	phone
Tax ID Number	taxnumber
VAT registration number	vatnumber
Home Page	website

The following fields are only used if Multiple Child Entities are `disabled`:

Business Component Field	Mapped Field
Primary Account Street Address	address1
Primary Account Address Street Address2	address2
Primary Account Address Street Address3	address3
Primary Account City	city
Primary Account State	adminarea

Business Component Field	Mapped Field
Primary Account Postal Code	`postalcode`
Primary Account Country	`country`

CUT Address - Data Cleansing

Business Component Field	Mapped Field
Street Address	`address1`
Street Address 2	`address2`
City	`city`
State	`adminarea`
Postal Code	`postalcode`
Country	`country`

CUT Address - DeDuplication

> **Note:**
>
> This Business Component is only used if Multiple Child Entities are `enabled`.

Business Component Field	Mapped Field
Street Address	`address1`
Street Address 2	`address2`
City	`city`
State	`adminarea`
Postal Code	`postalcode`
Country	`country`

Personal Address - Data Cleansing

> **Note:**
>
> This Business Component is only used if Multiple Child Entities are `enabled`.

Business Component Field	Mapped Field
Street Address	`address1`

Business Component Field	Mapped Field
Street Address 2	address2
City	city
State	adminarea
Postal Code	postalcode
Country	country

Personal Address - DeDuplication

> **Note:**
>
> This Business Component is only used if Multiple Child Entities are enabled.

Business Component Field	Mapped Field
Street Address	address1
Street Address 2	address2
City	city
State	adminarea
Postal Code	postalcode
Country	country

Contact - Data Cleansing

Business Component Field	Mapped Field
Id	individualid
M/M	title
M/F	gender
First Name	firstname
Middle Name	middlename
Last Name	lastname
Home Phone #	homephone
Work Phone #	workphone
Fax Phone #	faxphone
Cellular Phone #	mobilephone
Email Address	email
Job Title	jobtitle
Social Security Number	taxnumber

ORACLE

Business Component Field	Mapped Field
Birth Date	dob

Contact - DeDuplication

Business Component Field	Mapped Field
Id	individualid
M/M	title
M/F	gender
First Name	firstname
Middle Name	middlename
Last Name	lastname
Home Phone #	homephone
Work Phone #	workphone
Fax Phone #	faxphone
Cellular Phone #	mobilephone
Email Address	email
Job Title	jobtitle
Social Security Number	taxnumber
Birth Date	dob
Primary Account Name	accountname

The following fields are only used if Multiple Child Entities are disabled:

Business Component Field	Mapped Field
Primary Street Address	address1
Primary Address Street Address2	address2
Primary Address Street Address3	address3
Primary City	city
Primary State	adminarea
Primary Postal Code	postalcode
Primary Country	country

List Management Prospective Contact - Data Cleansing

Business Component Field	Mapped Field
Id	individualid
M/M	title

Business Component Field	Mapped Field
M/F	gender
First Name	firstname
Middle Name	middlename
Last Name	lastname
Home Phone #	homephone
Work Phone #	workphone
Fax Phone #	faxphone
Cellular Phone #	mobilephone
Email Address	email
Job Title	jobtitle
Social Security Number	taxnumber
Birth Date	dob

List Management Prospective Contact - DeDuplication

Business Component Field	Mapped Field
Id	individualid
M/M	title
M/F	gender
First Name	firstname
Middle Name	middlename
Last Name	lastname
Home Phone #	homephone
Work Phone #	workphone
Fax Phone #	faxphone
Cellular Phone #	mobilephone
Email Address	email
Job Title	jobtitle
Account	accountname
Social Security Number	taxnumber
Birth Date	dob

The following fields are only used if Multiple Child Entities are disabled:

Business Component Field	Mapped Field
Street Address	address1
Street Address 2	address2

Business Component Field	Mapped Field
City	`city`
State	`adminarea`
Postal Code	`postalcode`
Country	`country`

Understanding the Vendor Parameters

The following are vendor parameters:

Name	Value
Key Generation Process by Third Party	`Yes`, if using EDQ web services for key generation. `No`, if using Siebel Token expressions. **Note**: This parameter should be set to "Yes" if external key generation is supported by the installed version of Siebel (version 8.1.1.10 or later). If the parameter is set to `Yes`, the Query and Token Expression parameters will be ignored, as these will not control key generation or candidate selection for real-time matching. For earlier versions of Siebel this parameter must be set to `No` and the Query and Token Expression Parameters must be set.
Support Multiple Child Entities Deduplication	`Yes`, if matching using multiple child entities. Otherwise, `No`. **Note:** If this parameter is set to `Yes`, the CUT and Personal Address DeDup Record Type parameters are enabled. This parameter should only be set to `Yes` if Multiple Child Entities are supported by the installed version of Siebel (version 8.1.1.10 or later).
Account DataCleanse Record Type	`accountclean`
Account DeDup Record Type	`accountmatch`
CUT Address DataCleanse Record Type	`addressclean`
CUT Address DeDup Record Type	`addressmatch` **Note:** Only used if multiple child entities are enabled.
Personal Address DataCleanse Record Type	`addressclean`
Personal Address DeDup Record Type	`addressmatch` **Note:** Only used if multiple child entities are enabled.
Contact DataCleanse Record Type	`contactclean`
Contact DeDup Record Type	`contactmatch`
List Mgmt Prospective Contact DataCleanse Record Type	`prospectclean`

Name	Value
List Mgmt Prospective Contact DeDup Record Type	`prospectmatch`
Batch Max Num Of Records	200
Parameter 1	"session", "clusterlevel", "2" **Note:** There must be a space after each comma.
Parameter 2	"session", "matchthreshold", "70" **Note:** There must be a space after each comma.
Parameter 3	"session", "uselegacykeygen", "N" **Note:** There must be a space after each comma.
Parameter 4	"session", "keyprofileind", "Typical" **Note:** There must be a space after each comma.
Parameter 5	"session", "keyprofileent", "Typical" **Note:** There must be a space after each comma.

Query and Token Expression Parameters

These parameters should only be configured if keys are generated by Siebel, rather than by calling EDQ key generation services. This applies to versions of Siebel prior to 8.1.1.10.

If the installed version of Siebel supports the use of EDQ services for key generation, then these services should be used to ensure the right balance between performance and match effectiveness.

> **Note:**
>
> The values of these Query and Token parameters are provided as examples only. They assume that there is a good level of completion of address data for individuals, that postal code on the primary address, and that most individuals are assigned to accounts. If this is not the case for the available data, these parameters must be configured to use the most frequently populated identifying data fields. For more information, see *Oracle Siebel Data Quality Administration Guide*.

Name	Value
Account Query Expression	`"IfNull (Left ([Primary Account Postal Code], 5), '?????') + IfNull (Left ([Name], 1), '?') + IfNull (Mid([Street Address], FindNoneOf ([Street Address], '1234567890 '), 1), '?')"`
Account Token Expression	`"IfNull (Left ([Primary Account Postal Code], 5), '_____') + IfNull (Left ([Name], 1), '_') + IfNull (Mid([Street Address], FindNoneOf ([Street Address], '1234567890 '), 1), '_')"`

Name	Value
Contact Query Expression	`"IfNull (Left ([Postal Code], 5), '?????') + IfNull(Left ([Account], 1), '?') + IfNull (Left ([Last Name], 1), '?')"`
Contact Token Expression	`"IfNull (Left ([Postal Code], 5), '_____') + IfNull(Left ([Account], 1), '_') + IfNull (Left ([Last Name], 1), '_')"`
List Mgmt Prospective Contact Query Expression	`"IfNull (Left ([Postal Code], 5), '?????') + IfNull(Left ([Account], 1), '?') + IfNull (Left ([Last Name], 1), '?')"`
List Mgmt Prospective Contact Token Expression	`"IfNull (Left ([Postal Code], 5), '_____') + IfNull(Left ([Account], 1), '_') + IfNull (Left ([Last Name], 1), '_')"`

> **Note:**
>
> The preceding values in the query, token expressions and parameter names are case sensitive. Also, '_____' in the Account Token Expression and Contact Token Expression settings represents five underscore characters in a sequence.

Understanding the Connector Interface

For integrating EDQ with Siebel, three types of services are available. They are:

- Key Generation
- Matching
- Cleaning/Standardization

The data is sent to EDQ in a request and the fields that make up the request are defined by the vendor mappings configured in Siebel.

Listed below are some example XML results as passed back from the Siebel connector to Siebel. The 'id' field (Contact.Id, Account.Id or contactid in these examples) depends on the object type. The possible names for the fields are:

Table 3-1   Examples of XML results

OBJECT	REALTIME ID	BATCH ID
Account	Account.Id	accid
Contact	Contact.Id	contactid
Prospect	Prospect.Id	prospectid

For **Key Generation**, a response such as the following example (for Contacts) is returned. The responses for Prospects and Accounts differ only in that the Id attribute

is different (Prospect.Id for Prospects and Account.Id for Accounts) and the type of key values that may be returned.:

```
<Data>
  <KeyRecord>
    <Contact.Id>1-6HMMPYM</Contact.Id>
    <KeyValue>AD110FNL3GNL3^1298^VAN^LOU</KeyValue>
    <KeyPriority>42</KeyPriority>
  </KeyRecord>
  ....
</Data>
```

For **Matching**, a response such as the following example for Accounts is returned. Responses for Contacts or Prospects differ only in the Id attribute:

- Realtime match Example for Contacts:

```
<Data>
  <DuplicateRecord>
    <Account.Id>1-xe2</Account.Id>
    <DQ.MatchScore>98</DQ.MatchScore>
    <DQ.RuleName>N020 Full name exact, A010 Address exact, W030 Website
no data, P030 Phone no data, T030 Tax number no data, V030 VAT number
no data</DQ.RuleName>
    <DQ.RuleAttributes>NAME,ADDRESS</DQ.RuleAttributes>
  </DuplicateRecord>
</Data>
```

- Batch match Example for Contacts:

```
<Data>
  <ParentRecord>
    <DQ.MasterRecordsRowID>1-HAKB</DQ.MasterRecordsRowID>
    <DuplicateRecord>
      <contactid>1-HAJU</contactid>
      <DQ.MatchScore>87.0</DQ.MatchScore>
      <DQ.RuleName>N100 Standardized full name, A010 Address exact,
C070 Account name no data, D030 DOB no data, P030 Phone no data, E040
Email no data, I030 National ID number no data, T030 Tax number no
data</DQ.RuleName>
      <DQ.RuleAttributes>NAME,ADDRESS</DQ.RuleAttributes>
    </DuplicateRecord>
  </ParentRecord>
</Data>
```

The responses received from Cleaning/Standardization services are entirely defined by the vendor field mappings in Siebel. However, the fields returned in responses from the Key Generation and Matching services are fixed according to the fields that Siebel's DQ interface understands and can use. For more information, see Understanding the Field Mappings for Business Components.

> **Note:**
>
> The fields **DQ.RuleName** and **DQ.RuleAttributes** can only be used in Siebel 19.7 or later versions. Older Siebel versions ignore these fields.