Oracle Fusion Cloud Customer Experience

How do I get started with duplicate identification and resolution?

Oracle Fusion Cloud Customer Experience
How do I get started with duplicate identification and resolution?

G40459-01

Author: Crescentia David

Contents

Get Help

There are a number of ways to learn more about your product and interact with Oracle and other users.

Get Help in the Applications

Some application pages have help icons to give you access to contextual help. If you don't see any help icons on your page, click your user image or name in the global header and select Show Help Icons. If the page has contextual help, help icons will appear.

Get Support

You can get support at *My Oracle Support*. For accessible support, visit *Oracle Accessibility Learning and Support*.

Get Training

Increase your knowledge of Oracle Cloud by taking courses at *Oracle University*.

Join Our Community

Use *Cloud Customer Connect* to get information from industry experts at Oracle and in the partner community. You can join forums to connect with other customers, post questions, suggest *ideas* for product enhancements, and watch events.

Learn About Accessibility

For information about Oracle's commitment to accessibility, visit the *Oracle Accessibility Program*. Videos included in this guide are provided as a media alternative for text-based topics also available in this guide.

Share Your Feedback

We welcome your feedback about Oracle Applications user assistance. If you need clarification, find an error, or just want to tell us what you found helpful, we'd like to hear from you.

You can email your feedback to *oracle_fusion_applications_help_ww_grp@oracle.com*.

Thanks for helping us improve our user assistance!

1 Overview

Let's understand the concepts of duplicate identification and duplicate resolution.

Overview of Duplicate Identification

Duplicate identification is a process that identifies potential duplicate records for organization (account) and person (contact) records. You can use this process to identify duplicates in the following situations:

- Identify potential duplicate account and contact records in real-time when your salespeople create a customer record. This prevents the entry of duplicate records.
- Identify potential duplicates in batch mode for records already in the database or within the batch file itself.
- Identify duplicates during import of customer data for records already in the database or within the import batch file itself.

Duplicate identification uses the Enterprise Data Quality (EDQ) service. You require a separate license for the Oracle Fusion Data Quality Cloud Service.

Overview of Duplicate Resolution

Duplicate resolution is a set of processes that you can use, after duplicate records are identified, to consolidate those records. You can resolve duplicates in two ways, either by linking them or by merging them. Linking involves associating the duplicate records.

The linked records are treated as unique records in the data registry, and have their own unique identifiers. Merging involves combining duplicate records into one new master record.

The duplicate resolution functionality comes with tools and rules that you can use to determine whether to merge records or not. These rules can also help determine the record that should be kept as the golden master (the surviving record) and the record that should be deleted (the non-master record).

You can setup linking by configuring a couple of profile options discussed in the topic Duplicate Resolution Simplified Profile Options. However, the setup for merging is a bit more elaborate. It consists of configuring logic to:

1. Determine which record from a set of identified duplicates should be designated as the master record. You can set this up by configuring Set Master Record Rules.
2. Determine which attribute value instances from across the set of duplicates the master record should have. You can set this up by configuring Set Attribute Value Rules. The Set Master Record Rules and Set Attribute Value Rules are collectively called Survivorship Rules.
3. Determine whether the merge is violating any of the conditions under which a merge should be prohibited. You can set this up by configuring Agreement Rules.

You can easily setup survivorship and agreement rules in Application Composer using Groovy Script.

An alternative to using Groovy Scripts based survivorship and agreement rules is to configure them using Oracle Business Rules. You can do this configuration in the Setup and Maintenance work area using the Manage Survivorship Rules and Manage Agreement Rules setup tasks.

Note that if you're already using Oracle Business Rules, you can migrate from Oracle Business Rules to Groovy Script incrementally. For example, you could continue to use Oracle Business Rules to define agreement rules while using Groovy Script for your set master rules.

> **Note:** In case of Accounts, Customer Profile Quality Scoring is an additional method to resolve duplicates. This method helps in assigning scores to records and identifying a record with the highest score. This record with the highest score can be used as the single account record, instead of multiple duplicate records.

2 Duplicate Identification Setup

Duplicate Identification Setup

How do I identify duplicates?

Matching is a process that identifies the potential duplicates for account, contact, and address. You can identify the potential duplicate records in real-time when you create a customer record, and in batch mode for existing records.

Oracle provides the data matching solution for Oracle Cloud applications through integration of Oracle Enterprise Data Quality (EDQ). EDQ is a complete data quality product with capabilities such as profiling, standardization, matching and merging, which is also available as a standalone product.

Defining data matching involves setting up two components, matching and address cleansing. As part of implementing these components, you must perform the following related setup tasks in the Setup and Maintenance work area from the Data Quality Foundation functional area of the Customer Data Management offering:

- Manage Server Configurations
- Manage Enterprise Data Quality Matching Configurations

Note: You must set up the server configurations prior to implementing EDQ matching.

Enterprise Data Quality Server Configurations

Enterprise Data Quality (EDQ) server configurations are predefined configurations for EDQ integration. You can enable and disable a data quality management operation by selecting or deselecting the related EDQ server configuration.

EDQ Real-Time and Batch Basic Match Server is the predefined server configuration available for EDQ.

EDQ Real-Time and Batch Basic Match Server

Enable this configuration if you want to use the matching capabilities. Enabling this configuration lets you benefit from both real-time and batch matching features. Real-time matching is used to prevent entry of duplicate records. Batch data matching is used for identifying duplicates of existing records.

Enterprise Data Quality Matching Configurations

Enterprise Data Quality (EDQ) matching configurations comprise attributes and parameters for real-time and batch matching of entities to prevent duplicate entries and identify existing duplicates. EDQ real-time and batch matching are available for account and contact entities.

You have the option of using either the predefined ready-to-use configuration or copying and adapting it to your address matching requirements. The predefined EDQ matching configurations applicable for both real-time and batch matching are:

- Account Duplicate Identification
- Contact Duplicate Identification

These configurations are used to identify the duplicate account and contact entries. You can review and edit these predefined matching configurations to optimize the matching functionality to meet your needs.

EDQ Matching Process

In EDQ matching process, the record added or updated to the application for comparison is called a driver record. And, the records that are compared with the driver record are called the candidate records. Driver records are compared with each other, but candidate records are never compared with other candidates. The EDQ real-time matching process compares a single driver record against many candidates and returns possible duplicate records based on matching attributes and threshold. The batch matching process compares all driver records of the same type, such as account and contact, and identifies all possible matches within these sets of records.

The batch matching process runs in two modes, full batch and incremental batch. While the full batch mode matches all records against each other, the incremental mode matches a subset of records against all of their selected candidates. In batch matching, separate matching templates are provided that lets you specify different match rules. For example, you may want to minimize user intervention of adding customers in front end applications, and perform an exhaustive match on a regular basis.

The EDQ matching process for real-time and batch matching runs the EDQ Cluster Key Generation service and EDQ matching service for duplicate identification. The EDQ Cluster Key Generation service is called whenever a record is added or updated in an application. This service generates keys for records added as well as for the records that are updated in the application. These generated keys are stored in the application, which are then used to select the candidate records that may match to the data in the application.

The selected candidate records along with the driver record are returned to the EDQ matching service. Then, this service examines the records and decides which of the candidate records are a good match with the driving record. Once EDQ matching service arrives at the best match, it assigns a score to every duplicate record identified based on the strength of the match.

For more information about the EDQ matching process, see the Oracle Enterprise Data Quality Customer Data Services Pack Matching Guide at

http://docs.oracle.com/cd/E48549_01/doc.11117/e40737/toc.htm

Match Attributes

Match attributes define the attributes that are used for real-time and batch matching of the account and contact entities to identify duplicate entries. You use two types of attributes for matching:

- Match Identifier: Specifies the EDQ attribute that you want to use for matching
- Application Attributes: Specifies the application attribute that you want to use for matching

You can map the attributes in application with the corresponding EDQ attributes to create an attribute mapping. For example, for the Name EDQ attribute, you can select the Org.OrganizationName as the corresponding Organization attribute to create a mapping. You can define such attribute mappings for real-time matching, batch-data matching, or both.

When you map the attributes in the application with the corresponding EDQ attributes, you create a matching configuration setting for identifying duplicate entries. These settings are stored as matching keys in the application.

Whenever you change the attribute mappings, you must regenerate matching key values for the new or updated accounts and contacts. You can regenerate matching key values using the **Rebuild Keys** option in the Edit Matching Configuration page.

Match Configuration Parameters

Matching configuration parameters are system-level parameters that control aspects of the data quality matching services.

The following parameters control matching operations for identification of duplicate entries such as account and contact in the database, between database and sets of data, such as import batches, or within sets of data to resolve them from merging or linking.

Score Threshold

- Parameter Value: Between 0 and 100. Default Value: 90
- Parameter Description: Specifies the score above which the matched records are returned by the matching service. Records equal to or greater than the score are considered as matches and the records with scores less than the threshold are rejected.

Match Results Display Threshold

> **Note:** This match configuration parameter is enabled only for real-time matching.

- Parameter Value: Between 0 and 100. Default Value: 10
- Parameter Description: Controls the number of matched records that are returned by the real-time matching.

Preview Configuration

The Preview Configuration option lets you enter the following parameters to identify and view the duplicate matching records in real-time without rebuilding the keys.

- Cluster Key Level: Returns records based on the cluster key level. This parameter has three options:
  - Limited: helps to identify a unique record. Example: exact name , phone number, address, and postal code.
  - Typical: helps narrow down a record among many records. Example: address and city, name and postal code.
  - Exhaustive: helps loosely identify a record. Example: postal code.
- Score Threshold: Returns records based on score threshold.
- Maximum Candidates: Returns records based on maximum candidates.
- Match Results Display Threshold: Returns records based on the match results display threshold value.

Review Configuration Results

The Review Configuration Results option lets you check if the input account or contact entered for matching in the Edit Matching Configuration page returns the expected matched account or contact after the rebuilding of keys. Alternatively, in the Review Configuration Results page, you can enter the attribute information for one or more of the following matching configuration parameters that you want to match:

- Cluster Key Level: Returns records based on the cluster key level.

- Limited: helps to identify a unique record. You must specify an option that will uniquely identify a record. Here are some examples:
  - exact name
  - phone number

  You must specify either the exact name or the phone number.
- Typical: helps narrow down a record among many records. You must specify a combination of options that will narrow down a record. Note that using name alone isn't considered for matching. You must provide an additional value such as either email, phone number, or address to find matches. Here are some examples:
  - a combination of address and city
  - a combination of name and postal code
- Exhaustive: helps loosely identify a record. For example, postal code.

- Score Threshold: Returns records based on score threshold.
- Maximum Candidates: Returns records based on maximum candidates.
- Match Results Display Threshold: Returns records based on the match results display threshold value.

How You Manage Level of Indirection

You can control the level and number of indirect duplicates for a driver record using a user defined profile option ORA_ZCQ_LEVEL_OF_INDIRECTION. This profile option lets you include indirect duplicates. For example, look at a duplicate set having possible drivers A, C and candidates B, D as follows:

A - B

A - C

C - D

Here, we delete duplicate pair C-D because its winner C is a matched record of A-C. Hence, we lose D as a potential duplicate. This would possibly be identified as a duplicate only in the next batch run. However, we know that D is an indirect duplicate of A. If we set up the value of the profile option ORA_ZCQ_LEVEL_OF_INDIRECTION as 1, you can consider D as a matched record in the first batch run itself. Therefore, the duplicate sets would be as follows:

A - B

A - C

A - D (because D is now an indirect duplicate of A).

Let's understand how the profile option ORA_ZCQ_LEVEL_OF_INDIRECTION controls the level of indirect duplicates with another example where we have the duplicate pairs as A-B, A-C, C-D, D-E, and E-F. In this case, setting the profile option value as 1 would mean that only the first level of indirect duplicates which is C-D is considered as part of A's duplicate set, causing the A-D pair to be formed. However, if we set the profile option value as 2, it would also extend to second level of indirection. Therefore, A-D and also A-E would be the duplicate pairs identified because of the A-C, C-D, and D-E indirection sequence.

To control the level and thereby number of indirect duplicates for a driver record using the profile option ORA_ZCQ_LEVEL_OF_INDIRECTION , perform the following steps:

1. In the Setup and Maintenance work area, go to the **Manage Administrator Profile Values** task.
2. On the **Manage Administrator Profile Values** page, search for and select the profile option.

3. In the Profile Values section, click **Add**. A new row is added for you to specify the following conditions:
    - **Profile Level**: Specify the level at which the profile value is to be set. Select Site.
    - **Profile Value**: Select or enter the value, such as 1 or 2, depending on the required level of indirection.
4. Click **Save and Close**.

> **Note:** Changes in the profile values take effect for a user on the next sign in.

Related Topics

- How do I manage Enterprise Data Quality matching configurations?

How do I manage Enterprise Data Quality matching configurations?

You can manage Enterprise Data Quality (EDQ) matching configurations using the Manage Enterprise Data Quality Matching Configurations setup task.

You can perform the following tasks as part of managing Enterprise Data Quality (EDQ) matching configurations.

- Copying a predefined Enterprise Data Quality matching configuration
- Editing a copy of the predefined Enterprise Data Quality matching configuration

Copy a Predefined Enterprise Data Quality Matching Configuration

To copy or make a duplicate of a predefined Enterprise Data Quality matching configuration, complete these steps:

1. In the Setup and Maintenance work area, go to the following:
    - Offering: Customer Data Management
    - Functional Area: Data Quality Foundation
    - Task: Manage Enterprise Data Quality Matching Configurations
2. On the Manage Enterprise Data Quality Matching Configurations page, select the **Account Duplicate Identification** match configuration, and click **Duplicate**.
3. Click **Yes** in the Warning dialog box.
4. Click **Save** to save the copy of the predefined configuration.

> **Note:** You must save the copy of the predefined configuration, if you want to edit it.

5. Repeat Steps 2 to 4 to create copies of the predefined configuration for the Contact Duplicate Identification match configurations.

Edit a Copy of the Predefined Enterprise Data Quality Matching Configuration

To edit a copy of the predefined Enterprise Data Quality matching configuration, complete these steps:

> **Note:** You can't edit the predefined Enterprise Data Quality configuration. You can only make a copy of it and edit it by following this procedure.

1. In the Setup and Maintenance work area, go to the following:
    - Offering: Customer Data Management
    - Functional Area: Data Quality Foundation
    - Task: Manage Enterprise Data Quality Matching Configurations
2. On the Manage Enterprise Data Quality Matching Configurations page, select the copy of **Account Duplicate Identification** match configuration that you created in Copying a Predefined Enterprise Data Quality Matching Configuration section, and click **Edit**.
3. Select the **Active** check box to set the Account Duplicate Identification match configuration as the active configuration.

> **Note:** By default, the predefined configurations are always set to active. If there are one or more copies of the predefined configurations, then you can set any of them to active by selecting the Active check box. At any given point in time, only one configuration can be active.

> **Note:** Also, note that the Usage option is set to Both, which indicates that the configuration is for both real-time and batch matching.

> **Note:** You must rebuild keys before activating a new or copied configuration. For more information about Rebuilding Keys, see the Key Generation topic.

4. Click **Yes** in the Warning dialog box to set this configuration as active.
5. In the **Match Attributes** section, perform the following steps:
    a. Select a row to edit the mapping, and click the drop-down button for the selected row.
    b. Select the relevant attribute from the list.
    c. If the list doesn't display the attribute that you want for the mapping including published custom fields, then click **Search** to search for the attribute.

    > **Note:** To identify duplicates based on a custom field search the custom attribute in the Account attribute drop-down and then map it to any available unused EDQ attribute.

    d. Select the relevant option and click **OK**.
6. Select a **Cluster Key Level** option. For example, select **Typical**.
7. In the **Score Threshold** field, enter a value between 0 and 100, such as **85**. In the **Match Results Display Limit** field, enter a value between 0 and 100, such as **20**.

> **Note:** The Match Results Display Limit option isn't available for Batch.

8. Click **Save**.

> **Note:** You can see the Automerge Threshold and Autolink Threshold values in the Batch tab. You can change these values in the Manage Customer Hub Profile Options page or override these values while creating the duplicate identification batch criteria rules.

Related Topics

- Enterprise Data Quality Matching Configurations
- Automerge
- How do identify duplicates using key generation?
- Create Duplicate Identification Batches and Define Subset Rules
- Customer Hub Profile Options

Duplicate Identification Simplified Profile Options

How do I set up duplicate identification simplified profile options?

You can configure the duplicate identification simplified profile options in the Setup and Maintenance work area.

Navigate as follows:

- Offering: Customer Data Management
- Functional Area: Customer Hub
- Task: Manage Customer Data Management Options

Click the Duplicate Identification tab.

You can use these profiles options to specify:

Tuning

You can use this group of profile options to fine-tune the performance of your duplicate identification processes. The profile options under this category are:

- Enable Real-Time Key Generation
- Number of Batch Key Generation Jobs
- Number of Batch Matching Jobs
- Number of Real-Time Possible Duplicates

Advanced Options

You can use this set of profile options to configure diagnostic profile options that aren't related to the standard configuration activities. The profile options under this category are:

- Batch Duplicate Identification Debug Level
- Enable Fine Logging for Match Value Conversion

You can find more details about the duplicate identification simplified profile options in the individual topics about them.

Related Topics

- Enable Real-Time Key Generation
- Number of Batch Key Generation Jobs
- Number of Batch Matching Jobs
- Number of Real-Time Possible Duplicates
- Batch Duplicate Identification Debug Level
- Enable Fine Logging for Match Value Conversion
- How You Validate the Customer Data Management Setup

Enable Real-Time Key Generation

This setting controls whether CDM generates real-time business events for updating cluster keys whenever a customer record is created or modified. The underlying profile option is ORA_ZCQ_ENABLE_REALTIME_KEYGEN.

The options are:

- Yes: Specifies that the real-time business events trigger cluster key generation whenever a record is created or modified.
- No: Specifies that the real-time business events aren't triggered.

Consider these points when working with this setting:

- A recurring incremental batch key generation job is always required, regardless of this setting.
- Not all processes generate cluster key generation events, regardless of this setting. For example, file-based record creation and updating don’t generate key generation events.
- We recommend that you review the best practices before changing this setting listed in the Related Topics section.

Related Topics

- How do I set up duplicate identification simplified profile options?
- Number of Batch Key Generation Jobs
- Number of Batch Matching Jobs
- Number of Real-Time Possible Duplicates
- Batch Duplicate Identification Debug Level
- Enable Fine Logging for Match Value Conversion
- How You Validate the Customer Data Management Setup

Number of Batch Key Generation Jobs

This setting controls whether CDM specifies the maximum number of batch cluster key generation child jobs. The underlying profile option ZCQ_BATCH_KEYGEN_NUMJOBS.

Related Topics

- How do I set up duplicate identification simplified profile options?
- Enable Real-Time Key Generation
- Number of Batch Matching Jobs
- Number of Real-Time Possible Duplicates
- Batch Duplicate Identification Debug Level
- Enable Fine Logging for Match Value Conversion
- How You Validate the Customer Data Management Setup

Number of Batch Matching Jobs

This setting controls whether CDM specifies the maximum number of batch duplicate identification child jobs. The underlying profile option ZCQ_BATCH_MATCH_NUMJOBS.

Related Topics

- How do I set up duplicate identification simplified profile options?
- Enable Real-Time Key Generation
- Number of Batch Key Generation Jobs
- Number of Real-Time Possible Duplicates
- Batch Duplicate Identification Debug Level
- Enable Fine Logging for Match Value Conversion
- How You Validate the Customer Data Management Setup

Number of Real-Time Possible Duplicates

This setting controls the maximum number of possible duplicates that can be presented through a real-time interface, such as the Possible Duplicated popup window in Customer Center or the response to the REST findDuplicates API. The underlying profile option is ORA_ZCQ_MAX_CANDIDATES.

Before changing the Number of Real-Time Possible Duplicates setting, you must consider these points:

- The default value of 20 candidates is usually an optimal balance of performance, completeness, and usability.
- Increasing the number of returned candidates above the default value may affect the performance of real-time interfaces.
- Possible Duplicates are returned in descending order of their match score, thus the highest quality match candidates are presented first.
- Keep in mind how many possible duplicates your users are likely to review.

Related Topics

- How do I set up duplicate identification simplified profile options?
- Enable Real-Time Key Generation
- Number of Batch Key Generation Jobs
- Number of Batch Matching Jobs
- Batch Duplicate Identification Debug Level
- Enable Fine Logging for Match Value Conversion
- How You Validate the Customer Data Management Setup

Batch Duplicate Identification Debug Level

This setting controls whether CDM specifies the level of diagnostic logging for batch duplicate identification processes. The underlying profile option is ORA_ZCQ_BATCHMATCH_DEBUG_LEVEL. The options are:

- Debug: Specifies that the diagnostic logging for batch duplicate identification processes is debug.
- Informational: Specifies that the diagnostic logging for batch duplicate identification processes is collecting information.

Related Topics

- How do I set up duplicate identification simplified profile options?
- Enable Real-Time Key Generation
- Number of Batch Key Generation Jobs
- Number of Batch Matching Jobs
- Number of Real-Time Possible Duplicates
- Enable Fine Logging for Match Value Conversion
- How You Validate the Customer Data Management Setup

Enable Fine Logging for Match Value Conversion

This setting controls whether CDM enables fine logging for the data quality match value conversion process. The underlying profile option is ORA_ZCQ_CONVERTER_FINEST_LOGGING. The options are:

- Yes: Specifies that the fine logging for the data quality match value conversion process is enabled.
- No: Specifies that the fine logging for the data quality match value conversion process isn't enabled.

Related Topics

- How do I set up duplicate identification simplified profile options?
- Enable Real-Time Key Generation
- Number of Batch Key Generation Jobs
- Number of Batch Matching Jobs
- Number of Real-Time Possible Duplicates
- Batch Duplicate Identification Debug Level
- How You Validate the Customer Data Management Setup

How You Validate the Customer Data Management Setup

Customer Data Management comes with validation processes that you can run to validate whether your setup is complete and consistent. The validation process audits your setup and reports back whether dependent setup options are logically consistent.

For example, you may have specified that Groovy scripts must be used to select master records on the Duplicate Resolution Options tab. But you may have forgotten to provide the Groovy scripts. The validation process can identify such inconsistencies in the setup. Validations help reduce the number of issues resulting from improper setup and recommend the actions needed to fix the issues.

You can run the validation processes, after you have configured the duplicate identification and duplicate resolution simplified profile options, in the Setup and Maintenance work area using the following:

- Offering: Customer Data Management
- Functional Area: Customer Hub
- Task: Manage Customer Data Management Options

1. Click the Validate Setup tab.
2. Click the Run Validation Process button.

   The validation report displays:
   - The list of rules that were validated.
   - Their descriptions.
   - The severity of errors.
   - The recommended actions to fix them.

You can go back to the Duplicate Resolution Options or the Duplicate Identification Options tab to change the configurations as per the recommendations and rerun the validation on the Validate Setup tab until you are satisfied with the outcome.

Related Topics

- How do I set up duplicate identification simplified profile options?
- Enable Real-Time Key Generation
- Number of Batch Key Generation Jobs
- Number of Batch Matching Jobs
- Number of Real-Time Possible Duplicates
- Batch Duplicate Identification Debug Level
- Enable Fine Logging for Match Value Conversion

Unique Identifiers and Elimination Identifiers

The duplicate identification setup process lets you configure up to three unique identifiers and three elimination identifiers as part of your matching configuration.

Unique identifiers and elimination identifiers are special match tokens that you can use for cases where the value of a single attribute conclusively determines whether a set of records are duplicates, without needing to consider any other attribute values. The meanings of these Unique identifiers and elimination identifiers fields are as follows:

- Unique Identifiers - any records that have the same non-null value for an attribute that has been defined as a unique identifiers are automatically identified as duplicates, regardless of any other attribute values
- Elimination Identifiers - any records that have different non-null values for an attribute that has been defined as an elimination identifiers are never identified as duplicates, regardless of any other attribute values

Both unique identifiers and elimination identifiers only evaluate non-null values, and matching is still possible between records that have a mixture of populated and null elimination identifier or unique identifier values. For example, two records may be identified as duplicates when one record has a value in an elimination identifier field, the other record has no value (null) for the elimination identifier field, and the records also match on other attributes such as name and address.

To better understand the difference between unique identifiers and elimination identifiers, consider the following requirements:

- We will always consider contacts that have the same email address to be duplicates. (This requirement could be handled with a unique identifier field)
- We will never consider contacts that have different email addresses to be duplicates. (This requirement could be handled with an elimination identifier field)

If you choose to use Custom Scoring as part of your duplicate identification configuration, keep the following in mind:

- Records with different non-null elimination identifier values are never identified as duplicates, regardless of any scoring rules. Elimination identifier fields should not be included in custom scoring rules, and adding elimination identifier fields to your custom scoring rules may lead to unpredictable or inconsistent matching results.
- The custom scoring rules for unique identifier fields must always be defined as a single-value exact match rule with a score of 100. The standard matching configuration comes seeded with default custom scoring rules for the three available unique identifier fields, and these default custom scoring rules should not be modified. Unique identifier fields are specifically designed for their function and adding unique identifier fields to any other custom scoring rules may lead to unpredictable or inconsistent matching results.

Manage Custom Match Rules and Scoring

This topic describes how to enable the custom match rules and scoring functionality.

Use the following procedure to enable custom match rules for account and contact. Note that custom match rules are available only for account and contact.

1. In the Setup and Maintenance work area, go to the following:
    - Offering: Customer Data Management
    - Functional Area: Data Quality Foundation
    - Task: Manage Enterprise Data Quality Matching Configurations
2. On the Manage Enterprise Data Quality Matching Configurations page, drill down on the predefined or the user-defined matching configuration, for which you want to enable or disable custom match rules and scoring.
3. Scroll down to the **Scoring Type** drop-down list and select Custom from the Real-Time or Batch tabs on the Edit Match Configuration page.
4. Click **Save** or **Save and Close**

How do I create custom match rules?

You can use the custom match rule and scoring functionality to create your own match rules and scores based on your business requirements.

Use the following procedure to create custom match rules.

1. In the Setup and Maintenance work area, go to the following:
    - Offering: Customer Data Management
    - Functional Area: Data Quality Foundation
    - Task: Manage Enterprise Data Quality Matching Configurations
2. On the Manage Enterprise Data Quality Matching Configurations page, drill down on the predefined or the user-defined matching configuration for which you want to create or update predefined rules.
3. Click **Manage Match Rules** on the Edit Match configuration page.
4. On the Manage Match Rules page click New. Alternatively, you can click Duplicate to create a new rule based on an existing predefined or user-defined match rule.
5. Enter the following values for the newly created or copied match rule:
    - **Rule Name:** The name of the new rule. There's no restriction for the rule name.
    - **Match Rule Score:** The score that you assign to the rule. The value must be between 1 and 100.
    - **Rule Attribute:** The standard or predefined attributes that must be used in the custom match rules. Specify an attribute name followed by either Exact or Fuzzy to find exact matches or closer to the attribute value matches. You can specify any number of rules separated by commas but you must ensure that the rule is logical. Ensure that the syntax and spelling are exact. The following are examples:
        - `name:Exact,address:Exact,taxnumber:Exact`
        - `name:Exact,address:Fuzzy,taxnumber:Exact`

> **Note:** Address:Exact match returns a match when all three of the following match:
>
> - Address lines 1, 2, 3 and 4, postal code
> - Country
> - State or City
>
> City need not match exactly but the address is considered an exact match if the address lines fields, state, and country match.

6. Click **Save** or **Save and Close**.

Considerations for Selecting Sources of Match Rules and Scoring

The Customer Data Quality application comes with two options for match rules and scoring, predefined match rules (EDQ match rules) and custom match rules (Customer Data Management match rules).

You can use a drop-down button on the Enterprise Data Quality Matching Configuration page to select the match rules and scoring option best suited for your business requirements.

The predefined EDQ match rules include a single, view-only matching rule for each object such as account and contact. These match rules can't be edited, scores can't be changed, and no new rules can be added.

In case the predefined match rules aren't suitable for your business requirements, you should enable custom match rules and scoring. You should be able to turn on custom scoring by selecting Custom from the Scoring Type drop-down list on the Manage Enterprise Data Quality Matching Configurations UI page. You can use this functionality to create your own match rules and scores.

How You Configure Synonyms and Skip-words Dictionary

You can use synonyms and skip-words to improve the quality of match results. Synonyms help you to map words that are used interchangeably in your industry.

For example, Tech maybe mapped to Technology. Skip-words help reduce noise words such as titles like Mr or Ms in names. These synonyms and skip-words dictionaries are pre-defined as lookup types. You can configure these dictionaries by modifying the lookup types. These lookup types let you tailor the match results by word replacement patterns that may be unique to your application.

> **Note:** Synonyms are supported only for Person names, Organization Names, and Address lines 1 to 4.

You can add, edit, or remove words or patterns in the following out of the box lookup types:

- `ORA_ZCQ_CS_ACC_MAPPING_DATA`: Modify this lookup type to update the mapping of account data. For example, `Tech` indicates Technology.
- `ORA_ZCQ_CS_CON_MAPPING_DATA`: Modify this lookup type to update the mapping of contact data. For example, `Abbi` indicates Abigail.
- `ORA_ZCQ_CS_ADD_MAPPING_DATA`: Modify this lookup type to update the mapping of address data. For example, `Rd` indicates Road.

- `ORA_ZCQ_CS_ACC_STRIP_DATA`: Modify this lookup type to update the list of strip words for account data. For example, add `org` to the list to remove it during data matches.
- `ORA_ZCQ_CS_ADD_STRIP_DATA`: Modify this lookup type to update the list of strip words for address data.
- `ORA_ZCQ_CS_CON_STRIP_DATA`: Modify this lookup type to update the list of strip words for contact data. For example, add Mr. or Ms. to the list to remove them during data matches.

Note: These lookups don't support these special characters `! " # $ % ' * - : ; @`

You can access the lookup types by following these steps:

1. Sign in as a setup user such as, Sales Administrator, Master Data Management Application Administrator, or Application Implementation Consultant.
2. Click **Navigator** > **Setup and Maintenance** work area.
3. Click the Tasks menu and click Search. Search for **Manage Standard Lookups** task and open it.
4. Type `ORA_ZCQ_CS%` in the Lookup Type field and click Search.

   A list of lookup types is displayed.
5. Click to select a lookup type.

   The details of the lookup are displayed. You can add, delete, or modify the lookup values as per your requirement.

How do identify duplicates using key generation?

The EDQ matching process for real-time and batch matching makes use of the EDQ Cluster Key Generation service and the EDQ matching service for duplicate identification. Successful key generation is critical to duplicate identification.

Key generation identifies similar parties and assigns a key to each. When a matching configuration is made active, the application passes a set of keys (subset of parties) to the EDQ matching service to process for duplicate identification.

The EDQ Cluster Key Generation service must be run whenever a record is added or updated in the application. This service generates keys for records added as well as for the records that are updated in the application. If keys aren't generated, duplicate identification fails.

How You Configure Key Generation

The duplicate identification process uses matching keys, which must be maintained through a key generation process. You need to set up a recurring, incremental key generation job for each active data quality matching configuration for batch duplicate identification. You also may want to configure real-time key generation for immediate matching of new data as soon as new data is entered.

How You Setup Recurring Incremental Key Generation (Required)

You can schedule incremental key generation for an active matching configuration using the schedule key generation option on the Edit Matching Configuration page. This generates keys for records that don't have a key or if the key time stamp is older than that of the records. You must incrementally generate matching key values for the new or updated accounts and contacts.

To assure your duplicate identification processes use up-to-date matching keys, check if the following setup tasks have been completed.

1. In the Setup and Maintenance work area, go to the following:

- Offering: Customer Data Management
- Functional Area: Data Quality Foundation
- Task: Enterprise Data Quality Matching Configuration

2. Click the name of the required active match configuration.
3. Click Scheduled Key Generation.
4. Click Advanced.
5. Select Using a schedule option in the Advanced Options section.
6. Specify the frequency, the start date, and end date to run the scheduled process.
   You may want to select a distant end-date to reduce how often you must run this task and to help assure that the recurring schedule doesn't lapse.
7. Click Submit.
   The recurring incremental key generation scheduled process is submitted.

How You Specify Real-time Key Generation (Optional)

You may want to configure keys to be generated for immediate matching of new data as soon as new data is entered. For example, different users may be entering duplicate records at nearly the same time and you may want the keys to be generated immediately. Real-Time key generation process is available to handle such scenarios.

Follow these steps to enable real-time key generation:

1. Search and navigate to the Manage Administrator Profile Values task.
2. Search for the ORA_ZCQ_ENABLE_REALTIME_KEYGEN profile option.
3. Set the Site level profile option value to Yes.
4. Click Save and Close.

Best Practices for Real-Time Key generation

Here are guidelines to help you decide if real-time key generation is right for your business needs:

- Use Real-Time key generation only if an appropriately scheduled incremental key generation process doesn't support your business process needs.
- Certain processes, such as batch address cleansing, don't trigger real-time key generation events. Real-time key generation isn't sufficient to replace recurring incremental key generation, but it may be used as a complement for certain business scenarios.
- If possible, disable update events for higher-volume data creation and update processes, such as web service-based data integration flows.

To disable high-volume user accounts from generating update events:

1. Search and navigate to the Manage Administrator Profile Values task.
2. Search for the ZCA_PUBLIC_BUSINESS_EVENTS profile option.
3. Click **Actions** > **New** in the Profile Values section.
4. Specify the following values:
   - Set the Profile Level to User.
   - Select the User Name for which you want to suppress business events.
   - Set the Profile Value to No.
5. Click Save and Close.

How You Rebuild Keys

You must rebuild keys before activating a new configuration. You must rebuild keys if you change match configuration mappings or if you think that the keys are no longer valid because of updates to the records. You can regenerate matching key values using the Rebuild Keys option in the Edit Matching Configuration page.

How You Specify Real-time and Batch Key Generation Options

You can specify different key generation options for batch matching and real-time matching. Take for example the cluster key level parameter that has the values, limited, typical, or exhaustive. It's possible to select one value of this parameter, say limited, for batch matching and another, say exhaustive, for real-time matching, depending on how tightly you want the data quality engine to match records.

How You Review Key Generations Status

You can search for key generation jobs and review the status of each key generation job on the Manage Key Generation page. The following table describes the various possible key generation statuses for a matching configuration.

Key Generation Status	Description
Pending	Key generation for the configuration is required.
Processing	Key generation for the configuration is in progress.
Review Required	Key generation for this configuration needs review.
Ready	Key generation for this configuration is complete.

Select Customer Identifying URLs During Duplicate Identification

The Duplicate Identification process evaluates the domain name portion of URLs to help determine whether a set of customer records are likely to be duplicates of each other.

You can ensure that distinct customer records aren't incorrectly classified as duplicates because a common domain name is present in the web contact points collection when that domain name doesn't actually identify the organization. For example, customer records can be given web contact points based on social websites, analyst websites, or other types of web resources that use a common domain name when referencing the customer in a query string or page-level locator. By registering those common domain names, the duplicate identification process doesn't consider those web contact points.

You can use the Manage Standard Lookups setup task and create new entries under the ORA_ZCQ_FILTER_DOMAINS lookup type for the required URLS.

You can add the URL details as follows:

1. Sign in as a setup user such as, Sales Administrator, Master Data Management Application Administrator, or Application Implementation Consultant.
2. Click **Navigator** > **Setup and Maintenance** work area.
3. Click the Tasks menu and click Search. Search for Manage Standard Lookups task and open it.

4. Type ORA_ZCQ_FILTER_DOMAINS in the Lookup Type field and click Search.
   The details of the lookup are displayed.
5. In the Lookup Codes section, click **Action** > **New** and specify the URL details.
   Specify the domain name in the Lookup Code field for the lookup. For example, type TWITTER in the Lookup Code field. Also type the Meaning and Description for the lookup code.
6. Click **Save and Close**.

Identify Duplicates in Real-Time Using REST APIs and SOAP Web Services

This topic describes how to identify duplicates using REST APIs and SOAP Web services leveraging the Enterprise Data Quality (EDQ) engine.

Before you begin, perform these steps:

1. Enable **EDQ Real time and Batch Basic Match Server** in the Manage Data Quality Server Configurations page.
2. Identify the configuration code for the EDQ match configuration that you want to perform from the Setup and Maintenance work area by going to the following:
   - Offering: Customer Data Management
   - Functional Area: Data Quality Foundation
   - Task: Enterprise Data Quality Matching Configuration

How You Identify Duplicates Using REST APIs

To identify duplicates for accounts (organizations) in real-time using REST API, use the following URL:

```
crmRestAPI/resouces/latest/accounts/action/findDuplicates
```

To identify duplicates for contacts (persons) in real-time using REST API, use the following URL:

```
crmRestAPI/resouces/latest/contacts/action/findDuplicates
```

You must use the POST method of these APIs. For more information on findDuplicates, see the Identify Duplicate Accounts and Contacts section in the REST API for CX Sales and Fusion Service guide. You can use your proprietary, or a third party, REST API client to use the findDuplicates custom action to identify duplicates.

For more information on working with REST API client, see the Work with your REST Client topic in the REST API for CX Sales and Fusion Service guide.

How You Identify Duplicates Using SOAP Web Services (Deprecated)

To identify duplicates in real-time using SOAP Web services (deprecated), use the following URL:

```
https://servername/crmService/DQRealTimeService?WSDL
```

For more information on DQRealTimeService, see the Trading Community Real Time Data Quality section in the Oracle CX SOAP Web Services for CX Sales and Fusion Service guide.

You can use your proprietary, or a third party, SOAP Web services client to use the DQRealTimeService to identify duplicates. Enter the URL and run the relevant operations listed here with appropriate payloads:

- matchPerson: Use this operation to identify duplicate persons.
- matchLocation: Use this operation to identify duplicate locations.
- matchOrganization: Use this operation to identify duplicate organizations.

For more information on Invoking SOAP Web Services, see the Invoking SOAP Web Services chapter in the Oracle CX SOAP Web Services for CX Sales and Fusion Service guide.

FAQs for Duplicate Identification Setup

What's the difference between matching configurations and matching server configurations?

Matching configurations include parameters that can be set at the matching configuration level and modified depending on cleansing strategy, data, and result requirements. You can use these configurations during real-time matching to prevent duplicate entries and during batch matching to identify existing duplicates.

Matching server configurations provide the address and port of the data quality server used to process match requests. These configurations show both matching configuration and server configuration level parameters along with their type and cardinality. The parameters set at the server level are applicable to all the matching configurations.

What's the difference between real-time duplicate prevention and duplicate identification?

Real-time duplicate prevention identifies all possible duplicate records that may exist in the database for an entered record. This prevents entering of duplicate entities, such as organization, person, or location, into the database.

Duplicate identification identifies potential duplicate entities already existing in the database using batch matching, and resolves the actual duplicates by merging or linking.

3 Duplicate Resolution Setup

Duplicate Resolution Setup

How do I resolve duplicates?

Duplicate resolution is a set of processes that you can use, after duplicate records are identified, to consolidate those records.

You can resolve duplicates in two ways, either by linking them or by merging them. Linking involves associating the duplicate records.

The linked records are treated as unique records in the data registry, and have their own unique identifiers. Merging involves combining duplicate records into one new master record.

You can setup linking by configuring a couple of profile options discussed in the topic Duplicate Resolution Simplified Profile Options. However, the setup for merging is a bit more elaborate. It consists of configuring logic to:

- Determine which record from a set of identified duplicates should be designated as the master record. You can set this up by configuring Set Master Record Rules.
- Determine which attribute value instances from across the set of duplicates the master record should contain. You can set this up by configuring Set Attribute Value Rules. The Set Master Record Rules and Set Attribute Value Rules are collectively called Survivorship Rules.
- Determine whether the merge is violating any of the conditions under which a merge should be prohibited. You can set this up by configuring Agreement Rules.

You can use either use Groovy Scripts or Oracle Business Rules to define these configurations.

In case of Accounts, Customer Profile Quality Scoring is an additional method to resolve duplicates. This method helps in assigning scores to records and identifying a record with the highest score. This record with the highest score can be used as the single account record, instead of multiple duplicate records.

Customer Profile Quality Score

Assign a Customer Profile Quality Score to account records to help resolve duplicates without merging them.

You can identify and use the highest scoring account record as the primary or default account record.

You can use this method for resolving duplicates when records can't be deleted due to legal or other constraints. This method helps in identifying a potential record which can be used as the single account record instead of multiple duplicate records. All the child records for remaining duplicate records are mapped to the account record with the highest score.

This helps you to:

- Review, sort, and filter account records based on the completeness and quality of their profile records.
- Identify data quality issues and to rationalize data stewardship processes and resources.

- Prioritize and stage sales activities based on the confidence and quality of the customer profile data.

The score is a number between 1 and 100. It's calculated by selecting attributes and assigning them weights. These weights increase or decrease the effect of the attribute on the score. For example, a validated address is of better quality than a non-validated address. An address with validate = yes should improve the score rather than address with validate = no. If the attributes are populated with a value, you can specify the value. Similarly, the child attributes, their weights, and if required, their values must be specified. The final score is calculated as the ratio of the record's conferred score to the total possible score. In other words, the sum of satisfied parameters is divided by the sum of all parameters multiplied by 100.We recommend that you create a work sheet containing the attributes and their weights to test how they impact the score before they configure the scores in the application. Here's a sample work sheet:

Sample Worksheet

Parameter	Entity	Scoring Weight	Parameter is Satisfied?	Conferred Score
Address Exists	Address	10	1	10
Address is Validated	Address	20	0	0
Account has a phone number	Contact Point	10	1	10
Account has a name	Profile	10	1	10
		Total Possible Score = 50		Total Conferred Score =30

The score of the record is 20. This sheet will help you assign weights and test the scores of the records in the application.

How You Assign a Customer Profile Quality Score to an Account

Configure a score based on the completeness and attribute values of a customer profile.

You can then use a batch process to assign and maintain the score as an attribute of the account profile.

Configure and Activate a Scoring Model

1. Navigate to **Application Composer**.
2. Click the **Customer Profile Quality Configuration** link in the **Common Setup** area of the navigation panel. Customer Profile Quality Configuration screen is displayed.
3. Click **Create** and then click **Start**.
4. Enter a name and description for the configuration and click **Continue**.
5. Select the attributes that will determine the quality of the records. You can also assign weights for the attributes to increase or decrease their impact on the score. If you provide a value for the selected attributes, only records with the specified value are selected.
6. Similarly, select the child attributes, its weights, or values.
7. Review the details. Click **Save** to save the configuration and click **Activate** to activate this configuration. If you click Activate, the configuration that's currently active becomes inactive.

What to do next

- When you activate a customer profile quality score configuration, a recurring scheduled process with a daily frequency is automatically submitted to run the scoring job on your account records. To change the frequency of the scheduled process, use the standard Scheduled Process interface to define and submit a new Schedule Scoring Process job.
- You can create saved searches based on Profile Quality Score and sort the records. Note that the records with the highest scores are complete with better quality.

Duplicate Resolution Simplified Profile Options

How You Setup Duplicate Resolution Simplified Profile Options

You can configure the duplicate resolution simplified profile options in the Setup and Maintenance work area using the following:

- Offering: Customer Data Management
- Functional Area: Customer Hub
- Task: Manage Customer Data Management Options

You can use these profiles options to specify:

- Tuning

  You can use this group of profile options to fine-tune the performance of your duplicate resolution processes. These settings work together to provide multiple ways to optimize your performance based on your implementation scenario, project phase, and data shape. The profile options under this category are:

  - Maximum Concurrent Resolution Request Jobs Setting
  - Merge Mode Setting
  - Resolution Request Job Size Setting
  - Merge Scope Setting
  - Merge Request Notifications

- Duplicate Resolution Control

  Use the profile options belonging to this category to control the flow and level of automation of your duplicate resolution processes. These settings let you define where merge requests can originate and when data stewards need to be involved in duplicate resolution activities. The profile options under this category are:

  - Automerge Threshold Setting
  - Customer Center Merge Requests Setting
  - Default Resolution Type Setting
  - Data Steward Resolution Request Handling Setting
  - User Resolution Request Handling Setting

- Merge Behavior

You can use this set of profile options to configure how the merge process handles the records in a duplicate resolution request. These settings let you control the processing logic that governs the final outcome of a merge request, such as which record is retained as the master and how the process derives its final attributes. The profile options under this category are:

  - Agreement Rules Type Setting
  - Attribute Selection Type Setting
  - Enable Attribute Source Tracking Setting
  - Master Record Selection Setting
  - Merge Identical Child Records Setting
  - Add Groovy to Attribute Selection Setting

- Link Behavior

  You can use this set of profile options to configure how the link process handles the records in a duplicate resolution request. These settings let you control the processing logic that governs the final outcome of a link request, that is, which record is selected as the master. The profile options under this category are:
  - Main Record Selection Setting
  - Autolink Threshold Setting

You can find more details about the duplicate resolution simplified profile options in the individual topics about them.

Add Groovy to Attribute Selection Setting

Controls whether logic written with Application Composer Data Quality Rules are used to determine which attribute values are selected for the master record.

The underlying profile option is ZCH_GROOVY_RULE. The options are:

- No: Use only the option selected for the Attribute Selection Type setting to determine which attribute values are selected for the master record.
- Yes: Combine logic written with Application Composer Data Quality Rules with the option selected for the Attribute Selection Type setting to determine which attribute values are selected for the master record.

Consider these points:

- You can't combine Data Quality Rules with the Oracle business rules Attribute Selection Type setting option.

Agreement Rules Type Setting

Controls the method used to prevent records from being merged into other records incorrectly.

The underlying profile option is ZCH_AGREEMENT_TYPE. The options are:

- Default agreement rules: Only the seeded agreement rules are processed. These rules can't be edited or disabled.
- Default agreement rules with Oracle business rules: In addition to the default rules, merge incorporates logic written with Oracle business rules using the Manage Agreement Rules setup task.
- Default agreement rules with Data Quality Rules: In addition to the default rules, merge incorporates logic written with Application Composer Data Quality Rules.

Consider these points:

- You can't use Oracle business rules to configure agreement rules or Data Quality Rules to configure agreement rules, if you have set the Merge Scope to Customer data management specific areas with restrictions.

Attribute Selection Type Setting

Controls the method used to coalesce attribute values from the different records in a merge set onto the master record.

The underlying profile option is ZCH_SETATTRIBUTE_TYPE. The options are:

- No attribute survivorship rules selected: Merge only replaces null values on the master with non-null values from the duplicates.
- Use source confidence with oldest record as the tie breaker: In addition to replacing nulls with non-nulls, merge picks the attributes for the master based on the source confidence values configured using the Manage Source System Confidence setup task. In the case of multiple records in the duplicate set sharing the highest source confidence value for a given attribute, the value created at the earliest point in time is selected.
- Use source confidence with newest record as the tie breaker: In addition to replacing nulls with non-nulls, merge picks the attributes for the master based on the source confidence values configured using the Manage Source System Confidence setup task. In the case of multiple records in the duplicate set sharing the highest source confidence value for a given attribute, the value created at the most recent point in time is selected.
- Use Oracle business rules: In addition to replacing nulls with non-nulls, merge picks the attributes for the master based on logic configured with Oracle business rules using the Manage Survivorship Rules setup task.

Consider these points:

- The source confidence-based survivorship methods use optimized internal processes and may offer the best performance.
- The Attribute Selection Type can't be Oracle business rules if the Merge Scope is set to Customer data management specific areas with restrictions.

Autolink Threshold Setting

Defines the match score at or above which a duplicate resolution request of type Link is processed without requiring a data steward to review the request. The underlying profile option is ZCH_AUTO_LINK_THRESHOLD.

Consider these points:

- The highest possible match score is 100. So, if you set this value to 101, you can prevent any automatic Link processing.
- Use this setting only when the Default Duplicate Resolution type is Link.

Automerge Threshold Setting

Defines the match score at or above which a duplicate resolution request of type Merge is processed without requiring a data steward to review the request. The underlying profile option is ZCH_AUTO_MERGE_THRESHOLD.

Consider these points:

- The highest possible match score is 100. So, if you set this value to 101, you can prevent automatic Merge processing.
- Use this setting only when the Default Duplicate Resolution type is Merge.

Customer Center Merge Requests Setting

Controls whether you can submit merge requests directly from the Customer Center Account and Contact list pages.

Consider these points:

- If this option is enabled, the User Merge Handling setting controls whether or not those requests require a data steward's review before being processed.

Data Steward Resolution Request Handling Setting

You can select the Data Steward Resolution Request Handling setting to specify if a batch duplicate resolution request job automatically processes records or requires a review by a data steward.

The underlying profile option for this setting is ORA_ZCH_DS_MERGE_REQUESTS. The options for this setting are:

- Allow Processing Without Approval: batch duplicate resolution requests are automatically processed without any intervention from the data steward.
- Process Subject to Approval: Data steward reviews manually the batch duplicate resolution requests before processing.

Consider these points:

- Use the Create Resolution Request task in Party Center pages to create manual resolution requests.

Default Resolution Type Setting

Controls which duplicate resolution process type are assigned to new duplicate resolution requests. The underlying profile option is ZCH_DEDUP_REQUEST_TYPE_OPTION.

The options are:

- Merge: New duplicate resolution requests are queued for merging the duplicates into one master record.
- Link: New duplicate resolution requests are queued for linking the duplicates to each other without merging the records. The records to remain active.
- Generic: The data steward selects the duplicate resolution process type before processing the resolution requests.

Consider these points:

- To implement automatic duplicate resolution request processing, you must select either Merge or Link for the Default Duplicate Resolution setting.
- A data steward can change the duplicate resolution request type while reviewing a duplicate resolution request.

Enable Attribute Source Tracking Setting

Controls whether the attribute-level change by source system is tracked. This tracking is required for survivorship processes that use source confidence configuration to determine which attribute values in a duplicate set are ultimately written onto the master record. The underlying profile option is ZCH_ENABLE_SURVIVORSHIP.

Consider these points:

- The 'Use source confidence with oldest record as the tiebreaker' and 'Use source confidence with newest record as tiebreaker' attribute selection options require Attribute Source Tracking to be enabled.

- Attribute selection options that use Oracle Business Rules can be used without enabling Attribute Source Tracking but those rules can't access source confidence values.
- Once enabled, Enable Attribute Source Tracking can't be disabled because breaks in the source tracking history invalidate source confidence-based logic.

Main Record Selection Setting

Controls the method used to identify which record in a duplicate set becomes the main during a link. The underlying profile option is ORA_ZCH_LINK_SETMAIN_TYPE.

The options are:

- Main account is based on duplicate identification result: Internal logic of the duplication identification process determines the master.
- Select highest Customer Profile Quality Score as main: The record with the highest customer profile quality score is selected as the main record.
- Select the main record for link type using groovy scripts: Logic groovy script configured with Application Composer Data Quality Rules determines the main.
- Select the newest sales account as main: The record with the most recent creation date is the main.
- Select the oldest sales account as main: The record with the oldest creation date is the main.

Consider these points:

- The Select the older record as master and Select the latest record as master options use optimized internal processes and may offer the best performance.
- The Duplicate Identification batch process identifies a preliminary master record, but the final determination of master record occurs within duplicate resolution processing per the Master Record Selection Setting. Select the Select master based on duplicate identification results option if you want to retain the master record selection from the duplicate identification results.

Note: If the creation date value is exactly same for the records, then the party with the lowest party ID is chosen as the master record.

What are the Master Record Selection settings?

Controls the method used to identify which record in a duplicate set becomes the master during a merge.

The options are:

- Select master record using survivorship rule: Logic configured with Oracle business rules using the Manage Survivorship Rules setup task determines the master.
- Select the oldest record as master: The record with the earliest creation date is the master.
- Select the latest record as master: With the most recent creation date is the master.
- Select master based on duplicate identification results: Internal logic of the duplication identification process determines the master.
- Select master record using Data Quality Rules: Logic configured with Application Composer Data Quality Rules determines the master.

Consider these points:

- The Select the older record as master and Select the latest record as master options use optimized internal processes and may offer the best performance.

- Master Record Selection can't be survivorship rule or Data Quality Rules if the Merge Scope has been set to Customer data management specific areas with restrictions.
- The Duplicate Identification batch process identifies a preliminary master record, but the final determination of master record occurs within duplicate resolution processing per the Master Record Selection Setting. Select the Select master based on duplicate identification results option if you want to retain the master record selection from the duplicate identification results.

Note: If the creation date value is exactly same for the records, then the party with the lowest party ID is chosen as the master record.

Maximum Concurrent Resolution Request Jobs Setting

You can use the Maximum Concurrent Resolution Request Jobs setting to control the number of resolution request jobs that can be processed at any given time. If you don't set the maximum limit, all resolution request jobs are submitted for concurrent processing.

A greater number of concurrent resolution request jobs may clear a duplicate resolution request queue more quickly but may impact other processes and functions that use customer records. The underlying profile option is ORA_ZCH_MERGE_MAX_REQUEST_LIMIT.

Before specifying the limit for Maximum Concurrent Resolution Request Jobs, you must consider these points:

- The default value is 10 concurrent resolution request jobs.
- Increasing the number of concurrent resolution request jobs can be helpful during high-volume data initialization scenarios when processing the duplicate resolution queue is a high priority.
- The Maximum Concurrent Resolution Request Jobs setting and the Resolution Request Job Size setting work together to let you control the resolution request system better.

Merge Mode Setting

You can use the Merge Mode setting to select the optimized mode for merge to prevent the triggering of non-essential business events. This setting controls whether the merge uses preconfigured processing logic or workflows and generates integration events for Oracle Integration Cloud services.

Ideally, you should enable optimized mode for merge to improve application performance when:

- you're not integrating merge events with external systems.
- you don't need groovy scripts or object workflows to run when a master record is updated by a merge.

Merge Identical Child Records Setting

Controls whether the merge process merges or transfer certain types of child records when they have the same values. This setting currently controls the processing of addresses, phone contact points, and email contact points. The underlying profile options is ORA_ZCH_MERGE_IDENTICAL.

Merge Request Notifications

Controls whether notifications are sent when duplicate resolution requests are assigned and processed.

The options are:

- Send notifications for new requests: User created merge requests are created with the Send Notifications parameter set to Yes.
- Don't send notifications for new requests: User created merge requests are created with the Send Notifications parameter set to No.
- Don't send any notifications: Merge notifications aren't sent regardless of the Send Notifications parameter value on individual merge requests.

Consider these points:

- The Send Notifications parameter of merge requests created through the Duplicate Identification Batch process can be set at the Duplicate Identification Batch level.
- The Send Notifications parameter value of individual merge requests can be modified on the Duplicate Resolution list page.

Merge Scope Setting

You can use the Merge Scope setting to specify the business areas to be processed during a merge. This setting optimizes the size of the merge memory and execution profile and application performance. The underlying profile option is ORA_ZCH_MERGE_SCOPE.

You can specify any of these areas as the merge scope:

- Customer data management specific areas: Merges Customer Data Management foundation entities such as Organizations, People, Addresses, Phone Numbers, and Email Addresses.
- Customer data management specific areas with restrictions: Merges the same entities as the CDM scope. However, processing throughput is maximized by not invoking survivorship rules, integration events, or extended processing logic. You can use this option for high-volume data initialization scenarios.
- All customer relationship management related areas: Merges Customer Data Management foundation entities and transfers CRM transactional data, such as Opportunities, Activities, or Service Requests, from duplicate records to the master record.
- All functional areas: Merges Customer Data Management foundation entities and all merge-enabled entities on the point of deployment.

Before specifying the merge scope, you should consider these points:

- Select the appropriate merge scope to get the best performance.
- When you use Customer Data Management specific areas with restrictions merge scope, you must only select compatible options for other settings:
  - Master Record Selection can't be survivorship rule.
  - Attribute Selection Type can't be Oracle business rules.
  - Agreement Rules Type can't be Oracle business rules.

Resolution Request Job Size Setting

You can use the Resolution Request Job Size to specify the number of requests that each resolution request job can handle.

A greater number of resolution requests per resolution request job may increase resolution request processing throughput when the Merge Mode is optimized or the Merge Scope is set to Customer data management specific areas with restrictions. The underlying profile option is ORA_ZCH_MERGE_REQUEST_BATCH_SIZE_LIMIT.

Consider these points before specifying the resolution request job size:

- The default value is 100 resolution requests per resolution request job.
- Increasing the number of resolution requests per resolution request job can be helpful during high-volume data initialization scenarios when processing the duplicate resolution queue is a high priority.
- The Resolution Request Job Size setting and the Maximum Concurrent Resolution Request Jobs setting work together to let you have a fine-grained control of the resolution request system.
- Setting no value (null) for the Resolution Request Job Size setting distributes all pending resolution requests to the available resolution request jobs. If the ratio of resolution request to resolution request jobs is too high, system performance may be adversely affected.

User Resolution Request Handling Setting

Controls whether manually created resolution requests are processed automatically or require review by a data steward.

The underlying profile option is ZCH_USER_MERGE_REQUESTS. The options are:

- Allow Processing Without Approval: Manually created resolution requests are processed automatically.
- Process Subject To Approval: Data steward reviews manually created resolution requests before processing.

Consider these points:

- Use the Account and Contact list pages in Customer Center if Customer Center Merge Requests have been enabled to create manual merge requests or create requests automatically using duplicate resolution web services.

How You Validate the Customer Data Management Setup

Customer Data Management comes with validation processes that you can run to validate whether your setup is complete and consistent. The validation process audits your setup and reports back whether dependent setup options are logically consistent.

For example, you may have specified that Groovy scripts must be used to select master records on the Duplicate Resolution Options tab. But you may have forgotten to provide the Groovy scripts. The validation process can identify such inconsistencies in the setup. Validations help reduce the number of issues resulting from improper setup and recommend the actions needed to fix the issues.

You can run the validation processes, after you have configured the duplicate identification and duplicate resolution simplified profile options, in the Setup and Maintenance work area using the following:

- Offering: Customer Data Management
- Functional Area: Customer Hub
- Task: Manage Customer Data Management Options

1. Click the Validate Setup tab.
2. Click the Run Validation Process button.
   The validation report displays:
   - The list of rules that were validated.
   - Their descriptions.
   - The severity of errors.

- The recommended actions to fix them.

You can go back to the Duplicate Resolution Options or the Duplicate Identification Options tab to change the configurations as per the recommendations and rerun the validation on the Validate Setup tab until you are satisfied with the outcome.

Related Topics

- How do I set up duplicate identification simplified profile options?
- Enable Real-Time Key Generation
- Number of Batch Key Generation Jobs
- Number of Batch Matching Jobs
- Number of Real-Time Possible Duplicates
- Batch Duplicate Identification Debug Level
- Enable Fine Logging for Match Value Conversion

How do I run the Request Dispatch Job?

The Request Dispatch job is used to process duplicate resolution requests that are in Pending or Submitted status.

You can use the Run Request Dispatch Job setup task to run request dispatch jobs in the following modes:

- Basic: To submit the request dispatch job for immediate processing.
- Advanced: To schedule the process to run either immediately or on a recurring periodic schedule, such as every hour or every day.

  You can also submit request dispatch jobs directly from the Duplicate Resolution work area by clicking the Dispatch button, which will submit the job for immediate processing.

  You can use the Scheduled Processes page to monitor the status of Request Dispatch jobs by querying for "Schedule Duplicate Resolution Requests" processes or by querying the specific process ID that was provided when you submitted the job.

Set Up Duplicate Resolution Using Groovy Scripts

Overview of Duplicate Resolution Setup Using Groovy Script

Duplicate resolution is a set of processes that you can use, after duplicate records are identified, to consolidate those records. You can resolve duplicates in two ways, either by linking them or by merging them. Linking involves associating the duplicate records.

How you Configure Merges Using Groovy Scripts

A merge request consists of configuring logic to:

- Determine which record from a set of identified duplicates should be designated as the master record. You can set this up by configuring Set Master Record Rules.

- Determine which attribute value instances from across the set of duplicates the master record should contain. You can set this up by configuring Set Attribute Value Rules. The Set Master Record Rules and Set Attribute Value Rules are collectively called Survivorship Rules.
- Determine whether the merge is violating any of the conditions under which a merge should be prohibited. You can set this up by configuring Agreement Rules.

The linked records are treated as unique records in the data registry, and have their own unique identifiers. Merging involves combining duplicate records into one new master record.

You can setup duplicate resolution using Groovy scripts by configuring a couple of profile options discussed in the topic Duplicate Resolution Simplified Profile Options and specify the scripts in the Application Composer.

You can easily setup survivorship and agreement rules using Groovy Script in Application Composer. If you already have defined survivorship and agreement rules without using the Groovy Scripts, you can migrate them to Groovy Script incrementally. For example, you could continue to use your existing agreement rules while also using Groovy Script for your set master rules.

How You Enable Groovy Script-based Survivorship and Agreement Rules

Before you can start using groovy scripts to configure survivorship and agreement rules, you must enable groovy scripting in Setup and Maintenance.

Follow these steps to enable Groovy Script-based survivorship and agreement rules:

1. In the Setup and Maintenance work area, go to the following:
    - Offering: Customer Data Management
    - Functional Area: Customer Hub
    - Task: Manage Customer Data Management Options
2. In the Merge Behavior section under the Duplicate Resolution Options tab, select the groovy script options as the value for one or more of the following fields as required:
    - Master Record Selection: Select the Select master record using groovy scripts option to define the rules for selecting master records using Groovy Scripts.
    - Attribute Selection Type: To define attribute selection rules using Groovy Scripts, select the following options:
        - Select either Use source confidence with newest record as the tie breaker or Use source confidence with oldest record as the tiebreaker.
        - Select Yes for the Add Groovy to Attribute Selection field.
    - Agreement Rules Type: Select the Default agreement rules with groovy scripts option to define agreement rules using Groovy Scripts.

Create an Application Composer Sandbox

The configuration of Groovy Script survivorship and agreement rules is done in Application Composer using the standard Unified Sandbox framework for developing and testing your scripts.

We recommend that you create a separate, dedicated sandbox for survivorship and agreement rules rather than combine survivorship and agreement rule configuration with other types of Application Composer configuration activities. This approach gives you the greatest flexibility for iterative design, testing, and deployment of your survivorship and agreement rules.

To create an Application Composer Sandbox:

1. Click **Navigator** > **Configuration** > **Application Composer** > **Sandboxes**
2. Click Create Sandbox.
3. Specify a name and select Application Composer All Tools. Also select the Publishable option as Yes.
4. Click Create and Enter.
5. Click the Application Composer icon.
6. Go to Common Setup and click Data Quality Rules.

   On the Data Quality Rules page, you should can see six predefined templates for survivorship rules of the type set master and set attributes and agreement rules. You should can see three rules for the Contact (Person) entity and three for the Account (Organization) entity.

You're now ready to configure the survivorship or agreement rules using Groovy Scrip for accounts or contacts. For more information about creating sandboxes, see the Related Topics section.

Related Topics

- Overview of Sandboxes

Configure Predefined Data Quality Rules to Your Requirements in Application Composer

You can configure predefined data quality rules to your requirements in the application composer.

Follow these steps:

1. In your Sandbox dedicated for Data Quality rules, click the Application Composer icon.
2. Go to Common Setup and click Data Quality Rules to view the predefined data quality rules. You should be able to see six predefined templates for data quality rules, three for the Contact (Person) entity and three for the Account (Organization) entity. These templates are:
   - ContactSetMaster: Configure rules for determining the master record in contact merges.
   - ContactSetAttribute: Configure attribute survivorship rules for contact merges.
   - ContactMergeAgreement: Configure merge agreement rules for contact records.
   - AccountSetMaster: Configure rules for determining the master record in account merges.
   - AccountSetAttribute: Configure attribute survivorship rules for account merges.
   - AccountMergeAgreement: Configure merge agreement rules for account records.
3. To configure the merge processing logic for any of these templates, you can:
   - Click the required template.
   - Select the desired row and click **Actions** > **Edit**.
4. Create and save your scripts in the Groovy Script editing interface that's displayed when you click edit for a given survivorship or agreement rule.

Configure Groovy Script Based Set Master Record Rules

In this example, you will learn how to create a set master record rule to select the master record in a resolution request, with different logic depending on whether the resolution request is merge request.

The logical requirements of the scenario are as follows:

- Records integrated with RNOW source system are the top priority to be master record.
- If multiple records are present from a prioritized source system, use the most recently updated record as the tiebreaker.

- If no RNOW-integrated records are present in the merge, take whatever record had been designated as master by the upstream process.

Steps to Perform

1. Enable Groovy Scripts to select master records:
   a. In the Setup and Maintenance work area, go to the following:
      - Offering: Customer Data Management
      - Functional Area: Customer Hub
      - Task: Manage Customer Data Management Options
   b. In the Merge Behavior section, select the Select master record using groovy scripts option for the Master Record Selection field.
2. Create an Application Composer sandbox. Refer to the Create an Application Composer Sandbox topic for the steps.
3. Populate the sample script in the AccountSetMaster template:
   - Navigate to **Common Setup** and click **Data Quality Rules**.
   - Click AccountSetMaster.
   - Copy and paste the code given in the Sample Code section in the Edit Data Quality Rules page.
   - Click Save and Close.
4. Test the code. Refer to the Test the Survivorship and Agreement Rules Configuration topic to test the code.
5. Deploy the Code, after you're satisfied with the results of the Set Master Record Rules. See the topic: Deploy the Survivorship and Agreement Rules Configuration.

What the Sample Script Does

The script begins by calling the getRows() input function to access the records in the resolution request. After initializing additional variables, the script calls getAttribute("ResolutionType") to determine what type of resolution request is being processed.

If the Resolution Type is MERGE, the script loops through the records in the resolution request to inspect which source system reference assignments exist for each record. When a record with an RNOW source system reference is found, the row is added to the list of records having the given source system reference assignment.

Once all the rows in the resolution request have been tested for their source system reference values, the script tests whether any records from the prioritized source system reference were found. If records are found in the top priority list, they're tested by last updated date. Then the most recently updated record having the highest priority source system reference is designated to become the master record.

Finally, the script calls the selectMaster() output function to designate the master record. If a top-priority or second-priority record was identified earlier in the script, that record is provided to the selectMaster() function. If no priority record was identified, then the default master record specified by the upstream process will be retained as the master record for the resolution request.

Sample Code

```
try {
 def requestRows = getRows();
 def rowMaster = false;
 def osrMap = ['RNOW': []];
 def rowDefaultMaster = getMaster();
 def iHighestAccountScore = 0;
 def resolutionType = getAttribute("ResolutionType");

 if (resolutionType == "MERGE") {
```

```
for (row in requestRows) {
def osrRows = row.getAttribute("OriginalSystemReference");
osrRows.reset();
while (osrRows.hasNext()) {
def osrRow = osrRows.next()
if (osrRow.OrigSystem == "RNOW") {
osrMap['RNOW'].add(row);
}
}
}

if (rowMaster == false && osrMap['RNOW'].size() > 0) {
rowMaster = osrMap['RNOW'][0];
for (row in osrMap['RNOW']) {
if (row.LastUpdateDate > rowMaster.LastUpdateDate) {
rowMaster = row;
}
}
}
}

if (rowMaster) {
selectMaster(rowMaster);
} else {
selectMaster(rowDefaultMaster);
}

} catch (Exception e) {
def sMsg = "Exception in Account Set Master: " + e.getMessage();
println(sMsg);
}
```

Configure Groovy Script Based Set Attribute Value Rules

In this example, you will learn how to create a set attribute value rule to override the standard attribute source confidence-based survivorship processing with groovy script based on the classification code assignment of the records.

The logical requirements of the scenario are as follows:

- If the merge contains a record that has been classified as `OFN Category One account`, use that `OFN Category One record's value` for a set of key fields regardless of the attribute source confidence score.
- If multiple rows in the merge have been classified as `OFN Category One`, use the attribute values from the most recently updated row.

Steps to Perform

1. Enable Groovy Scripts to select master records:
    a. In the Setup and Maintenance work area, go to the following:
        - Offering: Customer Data Management
        - Functional Area: Customer Hub
        - Task: Manage Customer Data Management Options
    b. In the Merge Behavior section under the Duplicate Resolution Options tab, select the following options:
        - Select either Use source confidence with newest record as the tie breaker or Use source confidence with oldest record as the tiebreaker for the Attribute Selection Type field.
        - Select Yes for the Add Groovy to Attribute Selection field.
    c. Create an Application Composer sandbox. Refer to the Create an Application Composer Sandbox topic for the steps.
2. Populate the sample script in the `AccountSetAttribute` template:
    a. Navigate to **Common Setup** and click **Data Quality Rules**.
    b. Copy and paste the code given in Sample Code section in the Edit Data Quality Rules page.
    c. Click Save and Close.
3. Test the code. Refer to the Test the Survivorship and Agreement Rules Configuration section to test the code.
4. Deploy the code after you're satisfied with the results of the attribute value rules, you can. See the topic: Deploy the Survivorship and Agreement Rules Configuration.

What the Sample Script Does

The script begins by calling the `getRows()input function` to access the records in the merge request. Next, the script loops through each of the row records and accesses its Code Assignment collection. The script then loops through the code assignments to test whether the specified code assignment value is present. When a row having the specified code assignment is identified, the row is copied into an array of prioritized records for subsequent processing.

Once all the rows in the merge request have been tested for their code assignment values, the list of prioritized records is sorted based on the records' last update date. Finally, the script identifies the most recently updated row having the specified code assignment value. After this row is identified, the script calls the `selectAttribute() output function` to designate that priority row as being the attribute value source for a set of defined attributes.

Sample Code

```
try {
 def rowDuplicates = getRows();
 def rowPriorities = [];
 def exceptionAttributes = ['BusinessScope', 'CeoTitle'];
 def CAs;
 def CAi;
 for(row in rowDuplicates){
 CAs = row.getAttribute("CodeAssignment");
 if(CAs){
 for(CA in CAs){
 CA.reset();
 while(CA.hasNext()) {
 CAi = CA.next();
 if(CAi.ClassCategory == "OFN" && CAi.ClassCode == "OFN1") {
 rowPriorities.add(row);}
 }
 }
 }
 }
```

```
    if(rowPriorities.size()){
    def rowPriority = rowPriorities[0];
    for (row in rowPriorities){
    if (row.LastUpdateDate > rowPriority.LastUpdateDate){
    rowPriority = row;}
    }

    for (a in exceptionAttributes){
    selectAttribute(a, rowPriority);
    }
    }
  }

  catch(Exception e) {
    def sMsg = "Exception in Account Set Attribute: " + e.getMessage();
    println(sMsg);
  }
```

Configure Groovy Script Based Agreement Rules

In this example, we are operating in a complex business ecosystem where it's not always appropriate to merge certain accounts that have been identified as duplicates.

When the conditions that prohibit a merge from happening are encountered, the data steward needs to see an informative message explaining exactly which records blocked the merge, and for what reasons. The conditions that can prevent a merge are as follows:

When the conditions that prohibit a merge from happening are encountered, the data steward needs to see an informative message explaining exactly which records blocked the merge, and for what reasons. The conditions that can prevent a merge are as follows:

- A non-master record is integrated with the Legal Hold system
- A non-master record has a Certification Score value of 100

Steps to Perform

1. Enable Groovy Scripts to select master records:
    a. In the Setup and Maintenance work area, go to the following:
        - Offering: Customer Data Management
        - Functional Area: Customer Hub
        - Task: Manage Customer Data Management Options
    b. In the Merge Behavior section under the Duplicate Resolution Options tab, select the Default agreement rules with groovy scripts option for the Agreement Rules Type field.
2. Create an Application Composer sandbox. Refer to the Create an Application Composer Sandbox topic for the steps.
3. Populate the sample script in the `AccountMergeAgreement` template:
    a. Navigate to **Common Setup** and click **Data Quality Rules**.
    b. Click `AccountMergeAgreement`.
4. Copy and paste the code given in the Sample Code section in the Edit Data Quality Rules page.
5. Click Save and Close.
6. Test the code. Refer to the Test the Survivorship and Agreement Rules Configuration topic to test the code.
7. Deploy the code after you're satisfied with the results of the agreement rules. See the topic: Deploy the Survivorship and Agreement Rules Configuration.

What the Sample Script Does

The script begins by calling the `getNonMasters()input function` to access the non-master records in the merge request. Note that the result of a Set Master groovy scripts is expressed in `getMaster()` and `getNonMasters()` responses in Agreement Rule and Attribute Selection scripts. Next, the script loops through each of the row records to test for the two different conditions that would lead to the merge being rejected. The first test is to evaluate the source system reference assignments of the records to determine if the record is integrated with the Legal Hold system. The second test is to check the Certification Level value of the non-master records. If a given record matches either of the tests, a partial rejection message is added to the `vetoMessages` array.

After all the non-master rows in the merge request have been tested, the script checks to see if the `vetoMessages` array has any data in it. If it does, then a final rejection message is constructed from the data in the `vetoMessages` array and the merge is rejected with that constructed message being displayed to the data steward in the duplicate resolution UI.

Sample Code

```
try {
 def rowNonMasters = getNonMasters();
 def sMsg = "";
 def rejectMsg = "";
 def vetoMessages = [];
 for (row in rowNonMasters){
 def OSRs = row.getAttribute("OriginalSystemReference");
 OSRs.reset();
 sMsg = "";
 while(OSRs.hasNext()) {
 def OS = OSRs.next();
 if (OS.getAttribute("OrigSystem") == "LEGAL_HOLD" && sMsg == "") {
 sMsg = "A legal hold has been placed on Account " + row.getAttribute("PartyNumber");
 vetoMessages.add(sMsg);}
 }
 sMsg = "";
 if (row.getAttribute("CertificationLevel") == "100") {
 sMsg = "A Certification Level of 100 was found on Account " + row.getAttribute("PartyNumber");
 vetoMessages.add(sMsg);}
 }

 if (vetoMessages.size()) {
 rejectMsg = "Merge rejection reasons: ";
 vetoMessages.eachWithIndex { item, index ->
 rejectMsg += ((index +1) + ") " + item + " ");
 }
 rejectRequest(rejectMsg);
 }
}

catch(Exception e) {
 def sMsg = "Exception in Account Merge Agreement: " + e.getMessage();
 println(sMsg);
}
```

Test the Survivorship and Agreement Rules Configuration

You can test your Groovy Script-based survivorship and agreement rules configuration while working inside of the sandbox by creating a Test Merge Request in the Duplicate Resolution work area.

A Test Merge Request invokes whatever survivorship and agreement rules have been configured inside the current sandbox. For more information about creating test merge requests, see Related Topics section.

After you have identified potential duplicates in your database through a duplicate identification batch, you can resolve these duplicate sets by creating and submitting a duplicate resolution request.

To create and submit a resolution request:

1. Navigate to the Create Resolution Request UI page as follows: **Navigator** > **Customer Data Management** > **Duplicate Resolution** > **Tasks**
2. Click **Create Resolution Request**.
3. Search for and multi-select the duplicate records using the shift key.
4. Click Create Request.
5. Click Test Merge and click OK.

   You can optionally select one of the records as master. Once the request is submitted, the application generates a Request ID, which you can use to track the status of the duplicate resolution process.
6. You can tweak your survivorship and agreement rules configuration and retest your code using new test merge requests to ensure that the code is working as expected.

   > **Note:** This process tests the merge request configuration without changing your application data.

Deploy the Survivorship and Agreement Rules Configuration

When you're satisfied with your configured survivorship and agreement rules, you can deploy the configuration by using the standard sandbox publication process.

When you publish the sandbox, whatever Groovy Script-based rules were previously in the mainline configuration are replaced with whatever scripts are in the sandbox. Once the sandbox has been published, the Request Dispatch scheduled process begins to use the new Groovy Scripts during the processing of regular merge requests.

To publish a sandbox to deploy your survivorship and agreement rules configuration:

1. Click **Navigator** > **Configuration** > **Sandboxes**
2. On the Sandboxes page, click the name of the sandbox you want to publish.
3. Click Publish.

   > **Note:** The Publish button might be disabled for your sandbox because of various reasons. For example, you haven't yet made any changes in your sandbox, or the Control Publish Sandbox Action in Production Environment profile option (FND_ALLOW_PUBLISH_SANDBOX) is set to No.
4. Click Continue to Publish. The sandbox is published.
5. Click Done.

Best Practices for Configuring Groovy Scripts Based Survivorship and Agreement Rules

In this topic we discuss the best practices for configuring groovy scripts.

- If you configure your implementation to use Groovy Scripts for Set Master rules, merges aren't processed until a valid Set Master script has been deployed.
- Any survivorship rules written using the Manage Survivorship Rules setup task which uses the Oracle Business Rules framework, continues to function if you don't enable groovy script survivorship rules.
- For a given survivorship process type, such as Set Master or Agreement Rules, you can either use Groovy script or Oracle Business Rules. You can't combine the two frameworks for a single process type. For example you can't define one Set Master rule using Groovy Script and another Set Master rule using Oracle Business rules.
- You can combine Oracle Business Rules and Groovy script between different survivorship process types, such as using Oracle Business Rules for Set Master logic and Groovy script for Set Attribute logic.
- For best performance with attribute survivorship processing, try to use attribute source confidence as much as possible for your Set Attribute survivorship logic.

- Merge request processing may handle very large volumes of records, so groovy scripts should be as fast and efficient as possible. Avoid using `newView()` functions and web service calls unless absolutely necessary; and if necessary, assure that these types of operations will be fast and reliable.
- Select one of the Use source confidence Attribute Selection Type options from the Manage Customer Data Management Options setup page.
- If needed, use Groovy script along with your source confidence configuration to handle exception scenarios.

Overview of Groovy Scripting Functions

Groovy Script support for configuring survivorship and agreement rules is based on a specific set of functions that let you interact with the data records in the context of a merge request. These functions are of the following categories:

- Functions that let you inspect the records in the merge requests
- Functions that let you define the result of the merge request

These categories of specialized functions help you to create survivorship and agreement rules using standard Groovy Script syntax and operations.

Input Functions

These functions provide the data that your survivorship and agreement rules evaluate. Generally, these functions are called at the beginning of your script to instantiate the information required to determine the proper merge process outputs.

`getRuleType()`

This function lets you determine the functional context of the script. This function returns `SetAttribute`, `SetMaster`, or Agreement depending on which type of script calls it. It's generally not necessary to programmatically determine the rule type because the script types are presented as distinct functions within the Application Composer Data Quality Rules task. But there may be cases where it's helpful for logging or testing.

`getObjectType()`

This function lets you determine what type of party the merge request is processing. This function returns PERSON or ORGANIZATION depending on which type of script calls it. It's generally not necessary to programmatically determine the object type because the scripts for Persons and Organizations are clearly differentiated as distinct functions within the Application Composer Data Quality Rules task. But there may be cases where it's helpful for logging or testing.

`getMaster()`

This function lets you access the data record that has been identified as the master record for the merge request. The function is called without parameters and it returns a single Row object that contains the details of the master record. The following example shows a typical usage of this function:

```
def rowMaster = getMaster();
def masterName = rowMaster.getAttribute("OrganizationName");
// etc...
```

`getNonMasters()`

This function lets you access the set of data records that have been identified as the non-master records for the merge request because the merge process inactivates them. This function is called without parameters and it returns a list of row objects consisting of one list entry for each non-master record. It's important to note that the `getNonMasters` list isn't an `ADF recordset object`. `ADF recordset functions` such as reset() and first() don't work with the list. The following example shows a typical usage of this function:

```
getNonMasters()
def rowNonMasters = getNonMasters();
def nonmasterName;
```

```
for (nonmaster in rowNonMasters) {
nonmasterName = nonmaster.getAttribute("OrganizationName"); }
// etc...
```

getRows()

This function lets you access the full set of customer records for the merge request, which is the union of the master and non-master sets of rows. This function is called without parameters and it returns a list of row objects consisting of one list entry for each non-master record. Like the **getNonMasters** function, it's important to note that the **getRows** list isn't an **ADF recordset object. ADF recordset functions** such as reset() and first() don't work with the list. The following example shows a typical usage of this function:

```
def rowDuplicates = getRows()
 def duplicateName;
 for (duplicate in rowDuplicates) {
 duplicateName = duplicate.getAttribute("OrganizationName"); }
 // etc...
```

getSourceInfo(Row row, String attributeName)

This function lets you access information about which source system provided the current value of an attribute for a given master or non-master row. This function is called using the following parameters:

- A row object for the non-master or master row of interest
- The name of a source-confidence configured attribute

This function returns a source information record for the attribute in question. The structure of the source information record is as follows:

Attribute	Definition	Example
RecordId	The party ID of the person or organization record referenced by the row object parameter.	300100184760397
AttributeName	The name of the attribute parameter.	OrganizationName
AttributeValue	The current value of the attribute on the row.	Pinnacle Systems
Source	The code of the registered source system for the attribute value.	RNOW
SourceConfidenceLevel	The configured attribute source confidence value of the given attribute for the given source system.	90
SourceUpdateDate	The time stamp when the person or organization record was updated with the current value.	1/24/2020 11:48:03 PM

Note: The **getSourceInfo** function is only available for attributes that have been configured with source system confidence using the Manage Source System Confidence setup task.

The following example shows a typical usage of this function:

```
def rowDuplicates = getRows();
def rowSource;
def bestSource;
def bestValue;
bestSource = getSourceInfo(rowDuplicates[0],"OrganizationName");
for (row in rowDuplicates) {
rowSource = getSourceInfo(row,"OrganizationName");
if(rowSource.SourceConfidenceLevel > bestSource.SourceConfidenceLevel) {
bestSource = rowSource;
}
}
bestValue = bestSource.AttributeValue;
```

Output Functions

Output functions create the final behavior of the merge process based on the logic of a survivorship or agreement rule script. Generally, these functions are called at the end of the script after the data provided by the input functions has been evaluated with scripted logic.

selectMaster(Row row)

This function is used in Contact Set Master and Account Set Master scripts to specify which record from the merge request should be retained as the master record after the merge. This function takes a data Row instance as its only parameter, and whatever row is passed to the function is the record that's retained as the master. All other records in the merge request are inactivated during merge processing. The following example shows a typical usage of this function:

```
...
def masterRow = rowDuplicates[0];
for (row in rowDuplicates) {
if row.LastUpdateDate > masterRow.LastUpdateDate {
masterRow = row;
}
}
selectMaster(masterRow);
```

selectAttribute(String attributeName, Row row)

This function is used in Contact Set Attributes and Account Set Attributes scripts to define which attribute value instances from across the records in the merge should be used to build the master record. This function takes the name of an attribute and a Row instance as its parameters. The value for the given parameter that's found in the given row is retained on the master record. This function is logically equivalent to using the Duplicate Resolution override flow to select the source record for a given attribute. The following example shows the syntax of this function:

```
def rowDuplicates = getRows();
def bestSourceRow;
def fieldName = "OrganizationName";
rowBestSource = rowDuplicates[0];
for (row in rowDuplicates) {
if(getSourceInfo(row, fieldName).SourceConfidenceLevel > getSourceInfo(rowBestSource,
fieldName).SourceConfidenceLevel) {
rowBestSource = row;}
}
selectAttribute(fieldName, rowBestSource);
```

overrideAttribute(String attributeName, Object attributeValue)

This function is used in Contact Set Attribute and Account Set Attribute rules to specify an attribute value for the master record, which can't be derived in the normal fashion from the records in the merge request. This function takes the name of an attribute and the value for the attribute as its parameters and sets the final value of the master record's

given field to the given value. This function is logically equivalent to using the Duplicate Resolution Override flow to enter your own value for a given attribute.

Note: Ensure that the value's data type and format are correct because this function sets an externally-defined value.

The following example shows the syntax for this function:

```
def fieldName = "OrganizationName";
 def fieldValue = "Pinnacle Systems";
 overrideAttribute(fieldName, fieldValue);
```

This function is used in Contact Agreement Rule and Account Agreement Rule scripts to veto a merge request if a specified set of conditions are observed in the merge request's records. This function takes a single parameter which defines the rejection message that's displayed on the merge request if the rejection criteria are met. The following example shows the syntax for this function:

```
def rowNonMasters = getNonMasters;
 for (row in rowNonMasters) {
 if (row.value != null) {
 rejectRequest("Unable to merge contacts that have this value");}
 }
```

Evaluating the Data

Once you have called the appropriate functions, your survivorship or agreement rules script need to evaluate the data to determine the correct merge result. This evaluation process uses standard Groovy Script operators and functions. For more information about Groovy scripts, see the Oracle Applications Cloud Groovy Scripting Reference guide.

Putting It Together

You can generally follow this pattern in groovy scripting:

1. `Call Input Functions`
2. `Evaluate the Data`
3. `Call Output Functions`

To further illustrate this concept, the following is a simple script to determine the master record for a merge request based on the most recent Last Updated Date from the records:

```
/* Input Functions: call getRows() to initialize a list of the party records in the merge request and then
define a variable to designate the Master record and set it to the first record in the list of Rows */
 def rowDuplicates = getRows();
 def masterRow = rowDuplicates[0];
/* Evaluate the Data: iterate through the list of records to determine if the current list item
was more recently updated than whatever record has been designated the master. If the current record was
 more recently updated,
promote it to become the new Master */
 for (row in rowDuplicates) {
 if (row.LastUpdateDate > masterRow.LastUpdateDate) {
 masterRow = row;
 }
 }
/* Call Output Functions: use the selectMaster() function to dictate which record from the merge set
should become the master */
 selectMaster(masterRow);
```

Best Practices for Groovy Scripting

Consider the following points when planning and configuring your survivorship and agreement rules using groovy scripts:

- The Rows returned by the `getRows()`, `getNonMasters()`, and `getMaster()` functions is a standard Groovy Script list object, not an `Oracle ADF recordset object`. You must use standard Groovy methods for traversing the `recordset` such as for (item in list) instead of ADF functions such as `reset()`, `first()`, or `hasNext()`.
- The responses of the `getRows()`, `getNonMasters()`, or `getMaster()` functions are cached for each script execution. So the data state of row objects of your scripts don't show any changes within the scope of a script execution.
- The result of a Set Master script is reflected in the response to `getNonMasters()` or `getMaster()` functions called in Set Attribute or Agreement Rules scripts.
- You can't access the Resolution Request header object in your survivorship scripts. The only supported means for initializing data objects in your scripts are the `input functions` described in this topic.
- The `selectAttribute()` and `overrideAttribute()` functions can be used on top-level attributes of the Row object. Fields that contain embedded child record collections can't be manipulated with these functions.
- You can interact with custom attributes and custom child objects by using the API name for the attribute or object that was specified when the custom entity was created in Application Composer.
- The script fragments provided in this topic are intended to illustrate the syntax and usage of the `Input and Output functions`. Refer to the Sample Scripts section for examples of complete scripts.
- Some Best Practices for writing Groovy Scripts are available in the Performance Best Practices for Using Scripts section of Performance Best Practices for Extending Oracle CX Sales and Fusion Service (Doc ID 2170121.1) on My Oracle Support: https://support.oracle.com/epmos/faces/DocumentDisplay?id=2170121.1
- The Groovy Script survivorship and agreement rule templates should only be used to configure Set Master, Attribute Survivorship and Agreement rules. Use of these templates for general processing extension or automation isn't supported and may cause incorrect or unpredictable behavior.

How you Configure Link Using Groovy Scripts

Link requests consist of configuring logic to:

- Specify that Groovy script will manage linking.
- Enter the Groovy script in Application Composer.

How You Enable Groovy Script-based Linking

Before you can start using groovy scripts to configure link, you must enable groovy scripting in Setup and Maintenance.

Follow these steps to enable Groovy Script-based linking:

1. In the Setup and Maintenance work area, go to the following:
    - Offering: Customer Data Management
    - Functional Area: Customer Hub
    - Task: Manage Customer Data Management Options
2. In the Link Behavior section under the Duplicate Resolution Options tab, select the groovy script options as the value for the Main Record Selection setting.

Configure Predefined Data Quality Rules to Process Link Requests in Application Composer

The configuration of Groovy Script for linking is done in Application Composer using the standard Unified Sandbox framework for developing and testing your scripts.

We recommend that you create a separate, dedicated sandbox for linking rather than combine this configuration with other types of Application Composer configuration activities. This approach gives you the greatest flexibility for design and deployment of your linking configuration.

In this example, you will learn how to create a set main link record rule to select the main record in a link request:

- The record with the highest Customer Profile Quality Score should be selected as the main record.
- If multiple records are tied for the highest Customer Profile Quality Score, use the most recently updated record as the tiebreaker.
- If none of the records in the link request have been assigned a Customer Profile Quality Score, take whatever record had been designated as main by the upstream process.

To create an Application Composer Sandbox:

1. Click **Navigator** > **Configuration** > **Application Composer** > **Sandboxes**.
2. Click Create Sandbox.
3. Specify a name and select Application Composer under All Tools. Also select the Publishable option as Yes.
4. Click Create and Enter.
5. Click the Application Composer icon.
6. Go to Common Setup and click Data Quality Rules.
7. Click the AccountSetMainLink template.
8. Copy and paste the code given in Sample Code section in the Edit Data Quality Rules page.
9. Click Save and Close.

Deploy the code. See the topic: Deploy the Link Request Configuration.

What the sample script does

The script begins by calling the getRows() input function which returns all the Parties selected in the link request.

The script loops through the records in the resolution request to inspect each record's Customer Profile Quality Score value. Each record's value is compared to the highest value found so far, and if the current row has a higher value than the highest value found so far, that row is promoted to become the master record of the link set.

Finally, the script calls the selectMaster() output function to designate the main record. If a top-priority or second-priority record was identified earlier in the script, that record is provided to the selectMaster() function. If no priority record was identified, then the default master record specified by the upstream process will be retained as the master record for the resolution request.

Sample Script:

```
try {
def requestRows = getRows();
def rowMaster = false;
def rowDefaultMaster = getMaster();
def iHighestAccountScore = 0;
for (row in requestRows)
{
if (nvl(row.getAttribute("ProfileQualityScore"), 0) > iHighestAccountScore) {
rowMaster = row;
iHighestAccountScore = row.getAttribute("ProfileQualityScore");
}
}
if (rowMaster) {
```

```
    selectMaster(rowMaster);
    } else {
    selectMaster(rowDefaultMaster);
    }
    } catch (Exception e) {
    def sMsg = "Exception in Account Set Master: " + e.getMessage();
    println(sMsg);
   }
```

Deploy the Link Request Configuration

When you're satisfied with your configured link logic, you can deploy the configuration by using the standard sandbox publication process.

When you publish the sandbox, whatever Groovy Script-based rules were previously in the mainline configuration are replaced with whatever scripts are in the sandbox. Once the sandbox has been published, the Request Dispatch scheduled process begins to use the new Groovy Scripts during the processing of regular link requests.

To publish a sandbox to deploy your link management configuration:

1. Click **Navigator** > **Configuration** > **Sandboxes**.
2. On the Sandboxes page, click the name of the sandbox you want to publish.
3. Click Publish.

   > **Note:** The Publish button might be disabled for your sandbox because of various reasons. For example, you haven't yet made any changes in your sandbox, or the Control Publish Sandbox Action in Production Environment profile option (FND_ALLOW_PUBLISH_SANDBOX) is set to No.

4. Click Continue to Publish. The sandbox is published.
5. Click Done.

Best Practices for Configuring Groovy Scripts for Link Requests

In this topic we discuss the best practices for configuring groovy scripts.

- If you configure your implementation to use Groovy Scripts for linking, your duplicate resolution requests aren't processed until a valid link management script has been deployed.
- Link request processing may handle very large volumes of records, so groovy scripts should be as fast and efficient as possible. Avoid using newView() functions and web service calls unless absolutely necessary; and if necessary, assure that these types of operations will be fast and reliable.

Overview of Groovy Scripting Functions

Groovy Script support for configuring link requests is based on a specific set of functions that let you interact with the duplicate data records. These functions are of the following categories:

- Functions that let you inspect the records in the link requests
- Functions that let you define the result of the link request

These categories of specialized functions help you to process link requests using standard Groovy Script syntax and operations.

Input Functions

These functions provide the data that your logic evaluates. Generally, these functions are called at the beginning of your script to instantiate the information required to determine the proper link process outputs.

```
getObjectType()
```

This function lets you determine what type of party the link request is processing. This function returns PERSON or ORGANIZATION depending on which type of script calls it. It's generally not necessary to programmatically determine the object type because the scripts for Persons and Organizations are clearly differentiated as distinct functions within the Application Composer Data Quality Rules task. But there may be cases where it's helpful for logging or testing.

`getMaster()`

This function lets you access the data record that has been identified as the master record for the link request. The function is called without parameters and it returns a single Row object that contains the details of the master record. The following example shows a typical usage of this function:

```
Copy
 def rowMaster = getMaster();
 def masterName = rowMaster.getAttribute("OrganizationName");
 // etc...
```

`getNonMasters()`

This function lets you access the set of data records that have been identified as the non-master records for the link request. This function is called without parameters and it returns a list of row objects consisting of one list entry for each non-master record. It's important to note that the **`getNonMasters`** list isn't an **`ADF recordset object`**. **`ADF recordset functions`** such as reset() and first() don't work with the list. The following example shows a typical usage of this function:

```
Copy
 getNonMasters()
def rowNonMasters = getNonMasters();
 def nonmasterName;
 for (nonmaster in rowNonMasters) {
 nonmasterName = nonmaster.getAttribute("OrganizationName"); }
 // etc...
```

`getRows()`

This function lets you access the full set of customer records for the link request, which is the union of the master and non-master sets of rows. This function is called without parameters and it returns a list of row objects consisting of one list entry for each non-master record. Like the **`getNonMasters`** function, it's important to note that the **`getRows`** list isn't an **`ADF recordset object`**. **`ADF recordset functions`** such as reset() and first() don't work with the list. The following example shows a typical usage of this function:

```
Copy
def rowDuplicates = getRows()
 def duplicateName;
 for (duplicate in rowDuplicates) {
 duplicateName = duplicate.getAttribute("OrganizationName"); }
 // etc...
```

`getSourceInfo(Row row,String attributeName)`

This function lets you access information about which source system provided the current value of an attribute for a given master or non-master row. This function is called using the following parameters:

- A row object for the non-master or master row of interest
- The name of a source-confidence configured attribute

This function returns a source information record for the attribute in question. The structure of the source information record is as follows:

Attribute	Definition	Example
`RecordId`	The party ID of the person or organization record referenced by the row object parameter.	300100184760397
`AttributeName`	The name of the attribute parameter.	OrganizationName
`AttributeValue`	The current value of the attribute on the row.	Pinnacle Systems
`Source`	The code of the registered source system for the attribute value.	RNOW
`SourceConfidenceLevel`	The configured attribute source confidence value of the given attribute for the given source system.	90
`SourceUpdateDate`	The time stamp when the person or organization record was updated with the current value.	1/24/2020 11:48:03 PM

The following example shows a typical usage of this function:

```
Copy
 def rowDuplicates = getRows();
 def rowSource;
 def bestSource;
 def bestValue;
 bestSource = getSourceInfo(rowDuplicates[0],"OrganizationName");
 for (row in rowDuplicates) {
 rowSource = getSourceInfo(row,"OrganizationName");
 if(rowSource.SourceConfidenceLevel > bestSource.SourceConfidenceLevel) {
 bestSource = rowSource;
 }
 }
 bestValue = bestSource.AttributeValue;
```

Output Functions

Output functions create the final behavior of the based on the logic of a link process script. Generally, these functions are called at the end of the script after the data provided by the input functions has been evaluated with scripted logic.

`selectMaster(Row row)`

This function is used in Contact Set Master and Account Set Master scripts to specify which record from the link request should be selected as the master record. This function takes a data Row instance as its only parameter, and whatever row is passed to the function is the record that's retained as the master. The following example shows a typical usage of this function:

```
Copy
...
 def masterRow = rowDuplicates[0];
 for (row in rowDuplicates) {
 if row.LastUpdateDate > masterRow.LastUpdateDate {
 masterRow = row;
 }
 }
 selectMaster(masterRow);
```

Evaluating the Data

Once you have called the appropriate functions, your script needs to evaluate the data to determine the correct link result. This evaluation process uses standard Groovy Script operators and functions. For more information about Groovy scripts, see the Oracle Applications Cloud Groovy Scripting Reference guide.

Putting It Together

You can generally follow this pattern in groovy scripting:

1. Call Input Functions
2. Evaluate the Data
3. Call Output Functions

```
Copy
/* Input Functions: call getRows() to initialize a list of the party records in the merge request and then
define a variable to designate the Master record and set it to the first record in the list of Rows */
 def rowDuplicates = getRows();
 def masterRow = rowDuplicates[0];
/* Evaluate the Data: iterate through the list of records to determine if the current list item
was more recently updated than whatever record has been designated the master. If the current record was
 more recently updated,
promote it to become the new Master */
 for (row in rowDuplicates) {
 if (row.LastUpdateDate > masterRow.LastUpdateDate) {
 masterRow = row;
 }
 }
/* Call Output Functions: use the selectMaster() function to dictate which record from the merge set
should become the master */
 selectMaster(masterRow);
```

Best Practices for Groovy Scripting

Consider the following points when planning and configuring your survivorship and agreement rules using Groovy scripts:

- The Rows returned by the getRows(), getNonMasters(), and getMaster() functions is a standard Groovy Script list object, not an Oracle ADF record set object. You must use standard Groovy methods for traversing the record set such as for (item in list) instead of ADF functions such as reset(), first(), or hasNext().
- The responses of the getRows(), getNonMasters(), or getMaster() functions are cached for each script execution. So, the data state of row objects of your scripts don't show any changes within the scope of a script execution.
- You can't access the Resolution Request header object in your survivorship scripts. The only supported means for initializing data objects in your scripts are the input functions described in this topic.
- You can interact with custom attributes and custom child objects by using the API name for the attribute or object that was specified when the custom entity was created in Application Composer.
- The script fragments provided in this topic are intended to illustrate the syntax and usage of the Input and Output functions. See the Sample Scripts section for examples of complete scripts.
- Some Best Practices for writing Groovy Scripts are available in the Performance Best Practices for Using Scripts section of Performance Best Practices for Extending Oracle CX Sales and Fusion Service (Doc ID 2170121.1) on My Oracle Support: https://support.oracle.com/epmos/faces/DocumentDisplay?id=2170121.1
- The Groovy Script templates should only be used to configure Set Master. Use of these templates for general processing extension or automation isn't supported and might cause incorrect or unpredictable behavior.

Set Up Duplicate Resolution Using Oracle Business Rules

Overview of Duplicate Resolution Setup Using Oracle Business Rules

An alternative to using Groovy Scripts based survivorship and agreement rules is to configure them using Oracle Business Rules. Before you process merge requests, you must configure survivorship and agreement rules. You can do this configuration in the Setup and Maintenance work area using the Manage Survivorship Rules and Manage Agreement Rules setup tasks.

Set Up Survivorship

Survivorship Rules

Survivorship rules are a collection of business rules that determine the master or surviving record and its attributes during the merge operation.

Survivorship rules create the best version of a record from multiple source systems, based on business rules. You can configure survivorship rules to resolve conflicts while merging duplicate records.

Survivorship Rules Types

There are two types of survivorship rules. You can configure them based on your business needs. They are as follows:

- **Set master record**: Configure the set master record rule to define the criteria for selecting the master record from a set of potential duplicate records.
- **Set attribute value**: Configure the set attribute value rule to define the criteria for selecting the best attribute values from multiple input records.

Predefined Survivorship Rules

Six predefined set attribute value rules are provided ready-to-use with the application:

- **Least Recently Updated Organization Attribute (History Wins)**: This rule selects the organization attributes that have the oldest updated date.
- **Most Recently Updated Organization Attribute (Recent Wins)**: This rule selects the organization attributes that have the most recent updated date.
- **Highest Source Confidence Level Wins for Organization**: This rule selects the organization attribute values that have the highest source confidence.
- **Least Recently Updated Person Attribute (History Wins)**: This rule selects the person attributes that have the oldest updated date..
- **Most Recently Updated Person Attribute (Recent Wins)**: This rule selects the person attributes that have the most recent updated date.
- **Highest Source Confidence Level Wins for Person**: This rule selects the person attribute values that have the highest source confidence.

In addition, you can use predefined templates to create new Set Attribute Value rules.

To see these predefined attribute rules, click Search button on the Manage Survivorship Rules task. You can use these predefined survivorship rules as a starting point to define the criteria that's best for your business. These rules are updated with every release. You can also create, edit, and delete these rules. However, deleting an existing rule isn't

recommended. By default, these predefined survivorship rules are in the inactive status and you can activate these rules from the Manage Survivorship Rules task.

How You Enable Survivorship Rules

You can enable the survivorship functionality by setting the **ZCH_ENABLE_SURVIVORSHIP** profile option to Yes in the Setup and Maintenance work area, using the following:

- Offering: Customer Data Management
- Functional Area: Customer Hub
- Task: Manage Customer Hub Profile Options

> **Note:** While working with Survivorship Rules, check if you have the following roles and privileges:
>
> - ORA_ZCH_CDM_ADMIN_DUTY
> - ORA_CRM_EXTN_ROLE
> - ZCX_MANAGE_EXTENSIBLE_OBJECT_PRIV

How You Manage Survivorship Rules

You can create, edit, and delete survivorship rules in the Setup and Maintenance work area by going to the following:

- Offering: Customer Data Management
- Functional Area: Customer Hub
- Task: Manage Survivorship Rules

The rules use source system confidence level and other criteria to determine the attributes of the record that should be retained from a particular source system, and are stored in the survivorship rules dictionary XML file.

> **Note:** The application doesn't support changing survivorship rules inside the Application Composer sandbox. Therefore, the merge engine doesn't pick up the changes made to these rules inside the Application Composer sandbox. When you define custom attributes or custom objects in an Application Composer sandbox, you should Publish and Exit the sandbox before changing a survivorship rule in the Manage Survivorship Rules setup task.

Define Survivorship Rules

This example demonstrates how to create a survivorship rule. Survivorship rules enable intelligent creation of the best version record, especially from multiple source systems, by specifying criteria for selecting the record to be retained during a merge operation.

Create A Survivorship Rule

To create a survivorship rule:

1. In the Setup and Maintenance work area, go to the following:
    - Offering: Customer Data Management
    - Functional Area: Customer Hub
    - Task: Manage Survivorship Rules

2. On the Manage Survivorship Rules page, click **Add** from the Actions menu.
3. Enter the sample information provided in the following table on the Create Survivorship Rule page.

Field	Value
Rule Name	PickPersonMasterRule
Description	Select the master person record based on original source system of the record.
Rule Type	Set master record **Note:** Note: You can create the following two types of survivorship rules: Set Master Record and Set Attribute Value. You can use predefined templates to create the Set Attribute Value rules.
Object Type	Person **Note:** You can create a survivorship rule for the following two types of party records: Person and Organization.

4. Click **Apply**. The Define Survivorship Rules: Select Master Record page appears.

Specify Criteria for Selecting the Master Record

The following are the steps to specify criteria for selecting the master record:

1. Navigate to the Define Survivorship Rules: Select Master Record page.
2. Enter the information provided in the following table as IF/THEN rules condition in the Define Survivorship Rules: Select Master Record page.

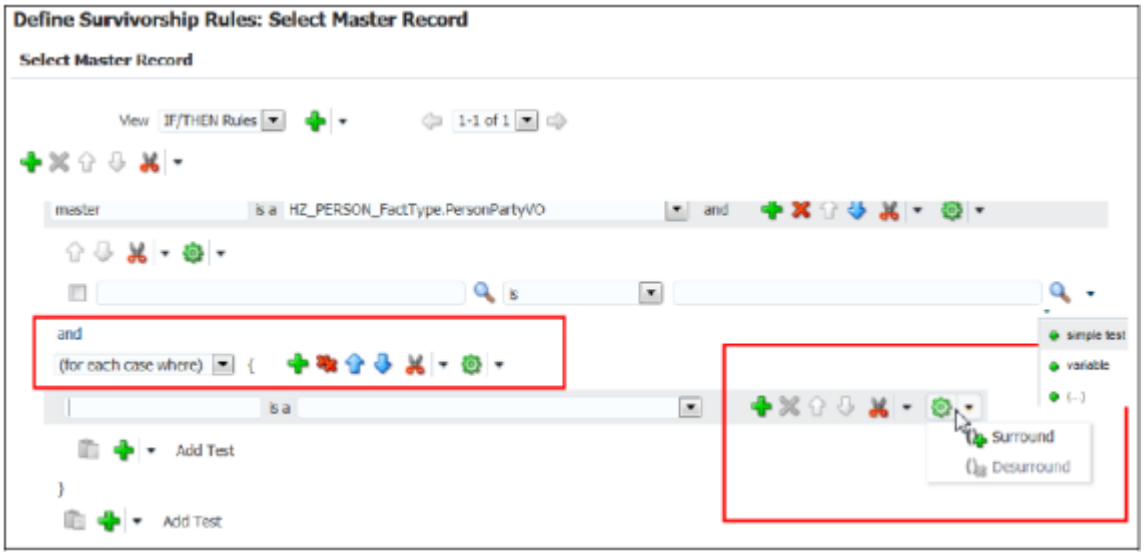
Rule Condition	Value
IF Condition	`IF PersonParty is a TCHFactTypeDictionary.PersonPartyVO and there's a case where {OrigSourceSystem} != null and OrigSourceSystem.OwnerTableId == PersonParty.PartyId and OrigSourceSystem.OrigSystem == "GSI"}`
THEN Condition	`THEN Assert new Result (name:"masterId", value:"PersonParty.PartyId)`

Note the following specification in the Define Survivorship Rules: Select Master Record page.

- Click the + icon to add additional patterns to include additional conditions
- Click the Surround with Parenthesis option to add more features to the conditions

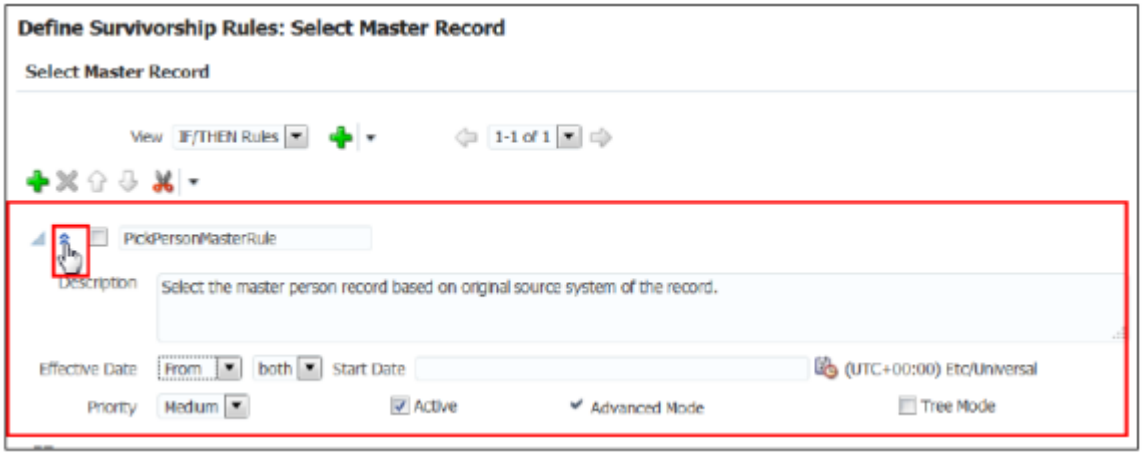
- Select the + Simple Test option to add additional clauses

The following figure shows the Define Survivorship Rules: Select Master Record page with the Surround icon highlighted.



3. Click the Advanced Settings button to verify the effective date of the rule that you're going to create.

   The following figure shows Advanced Settings icon on the Define Survivorship Rule: Select Master Record page.

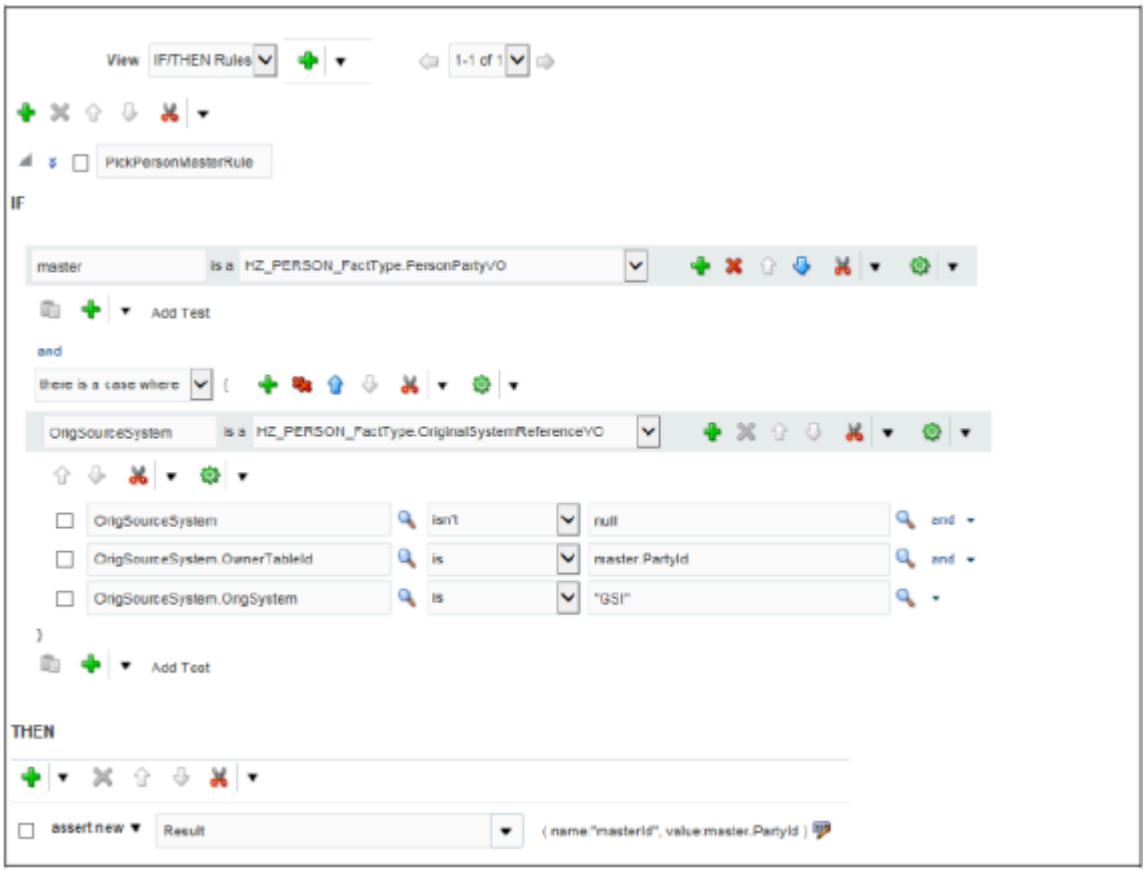


4. Select an effective date, as appropriate. The default effective date is Always. You can select an Effective From or Effective To date, or you can select an Effective Date range.
5. Select a priority for the rule, as appropriate.

   The default priority of the Survivorship rules is Medium. These rules get executed in the order of their priority.

6. Ensure that the Rules Active option is selected.

   The following figure shows how the survivorship rule looks like when fully defined. The details include the rule name and the IF and THEN conditions for determining the master.



7. Click Save and Close. You can view the newly created rule in the Manage Survivorship Rules page by searching for it.
8. Click Submit.

> **Tip:** To activate the rule you must click Submit. You may have selected the Active mode, but that doesn't activate a rule unless submitted.

Related Topics

- MOS document: Define Survivorship Rules

Define Set Master Record Rules

This procedure demonstrates how to create a survivorship rule of the type Set Master Record.

You can determine survivorship at the record level using the set master record rule type. Set master rules are used in party merge to set a single record as the master record.

Create Set Master Record Rules

The input to the survivorship rule is given in the IF clause. In a set master rule, the input is a set of party records. The THEN clause contains the output that determines the master record. In the Set Master Record rule, the output is a

result object that contains a specific Party ID. If multiple records with different Party IDs are returned, then it results in a conflict error. To create Set Master rules, you may perform the following steps:

To create Set Master rules, you perform the following steps:

1. Navigate to the Manage Survivorship Rules task.
2. Click Add. The Create Survivorship Rule page appears.
3. Enter the information provided in the following table on the Create Survivorship Rule page.

Field	Value
Rule Name	PickOrganizationMasterRule
Description	Select the master organization record based on the specified criteria for setting the master record.
Rule Type	Set master record
Object Type	Organization

4. Click Apply. You're taken to the Define Survivorship Rules: Select Master Record page.

In the Define Survivorship Rules: Select Master Record page, you specify criteria for picking the master record. The criteria that you define in this page determine the output of the rule.

The following topics contains three worked examples that show different ways of defining criteria in the Define Survivorship Rules: Select Master Record page to set a master records:

- Set the Record with Oldest Creation Date as Master
- Set the Record with D-U-N-S Number and Smallest Party ID as Master
- Set the Record with D-U-N-S Number and Highest Number of Party Site as Master

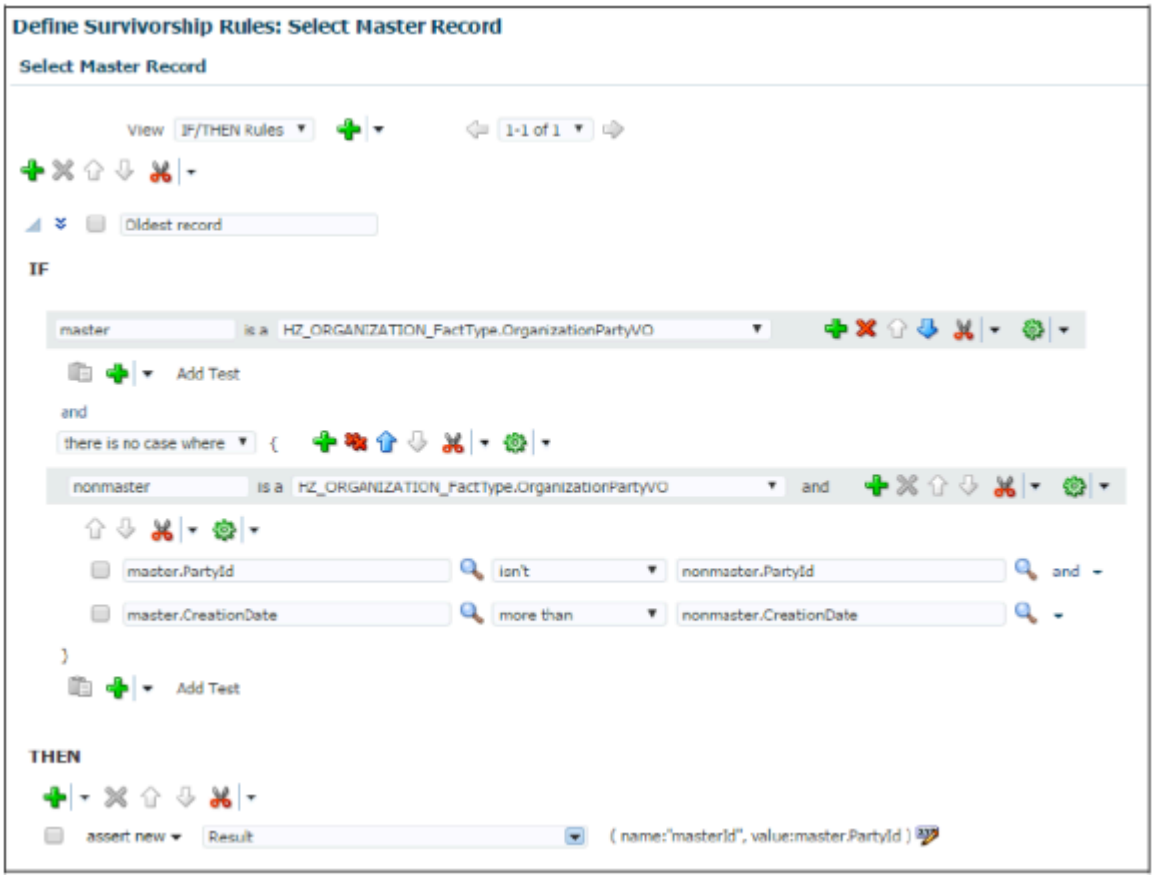
How You Set the Record with the Oldest Creation Date as Master

This rule has a single condition to set a record that has the oldest creation date as the master.

1. Navigate to the Define Survivorship Rules: Select Master Record page.
2. Enter the information provided in the following table as IF/THEN rules condition in the Define Survivorship Rule: Select Master Record page.

Rule Condition	Value
IF Condition	`IF master is an HZ_PERSON_FactType.PersonPartyVO and there is no case where {nonmaster == HZ_PERSON_FactType.PersonPartyVO and master.PartyId isn't nonmaster.PartyId and master.creationDate is more than nonmaster.CreationDate}`
THEN Condition	`THEN Assert new Result (name:"masterId", value:" master.PartyId)`

The following figure displays the Define Survivorship Rules: Select Master Record page with completely filled IF and THEN rules conditions for setting a record that has the oldest creation date as the master.



How You Set the Record with D-U-N-S Number and Smallest Party ID as Master

This rule identifies and returns the master record based on the following three conditions in the order of priority listed:

1. Pick master that has D-U-N-S Number.
2. If more than one record has D-U-N-S Number, pick one based on the smallest Party ID.
3. If no record has D-U-N-S Number, pick one based on the smallest Party ID.

The following are the use cases for a set master record rule to pick the master based on the D-U-N-S number and the smallest Party ID.

Use Case 1

In this case, there are two records with D-U-N-S number. Therefore, the record with the smaller party ID is picked as the master record. The following table contains the sample record information for this use case.

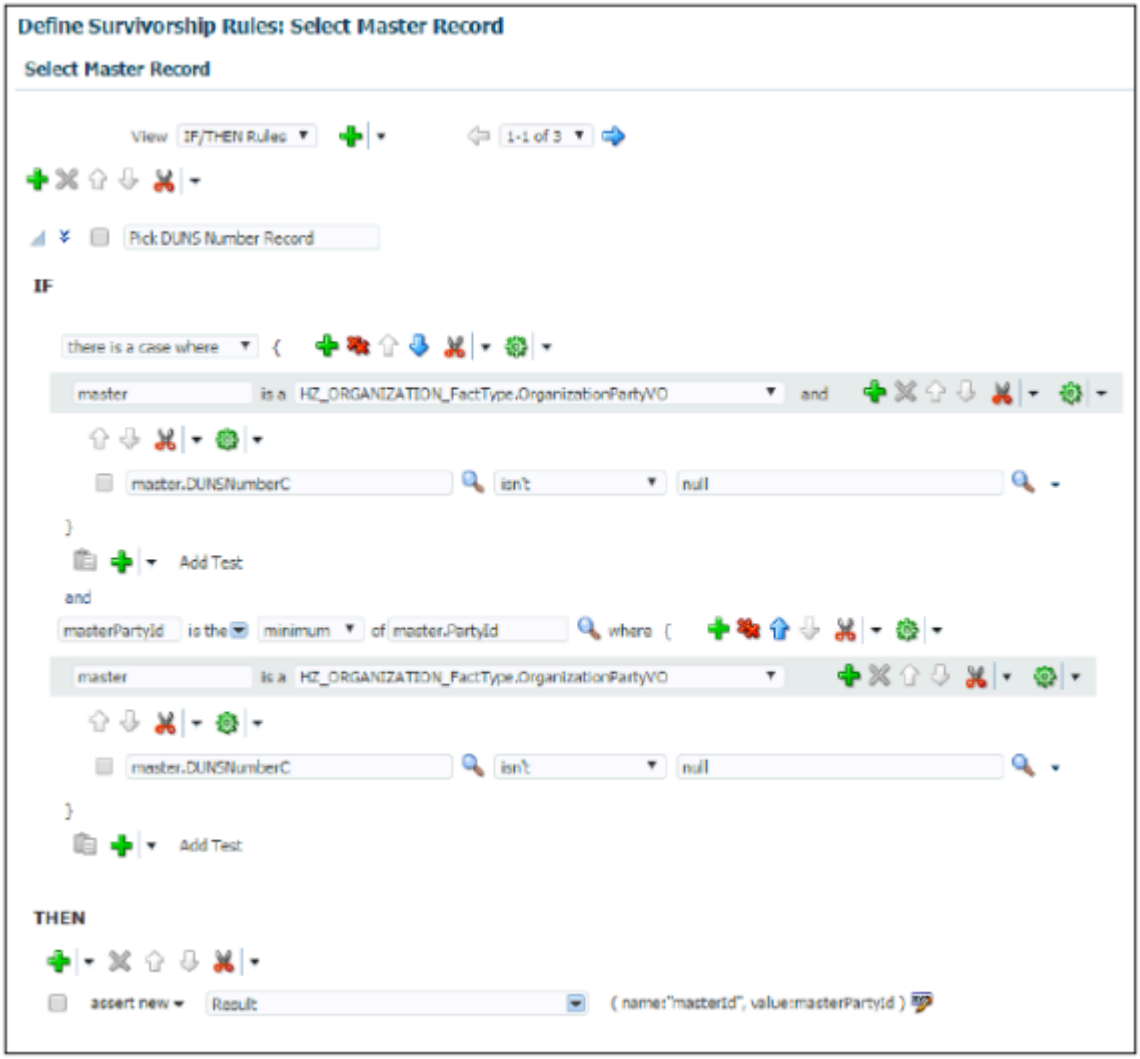
Record Name	Party ID	D-U-N-S Number	Master
Record 1	11	998837472	Yes
Record 2	12	null	No
Record 3	13	984939234	No

Record Name	Party ID	D-U-N-S Number	Master

The following table lists the IF and THEN rules condition values that you must enter on the Define Survivorship Rules: Select Master Record page for this use case.

Rule Condition	Value
IF Condition	`Pick D-U-N-S number{IF master is an HZ_PERSON_FactType.OrganizationPartyVO master.DUNsNumberC isn't null} masterPartyID is the minimum of masterPartyID where {master= HZ_PERSON_FactType.OrganizationPartyVO and master.DUNsNumberC isn't null}`
THEN Condition	`THEN Assert new Result (name:"masterId", value:" master.PartyId)`

The following figure shows the Define Survivorship Rules: Select Master Record page with IF and THEN rules conditions for picking the record with D-U-N-S number and minimum party ID as master.



Use Case 2

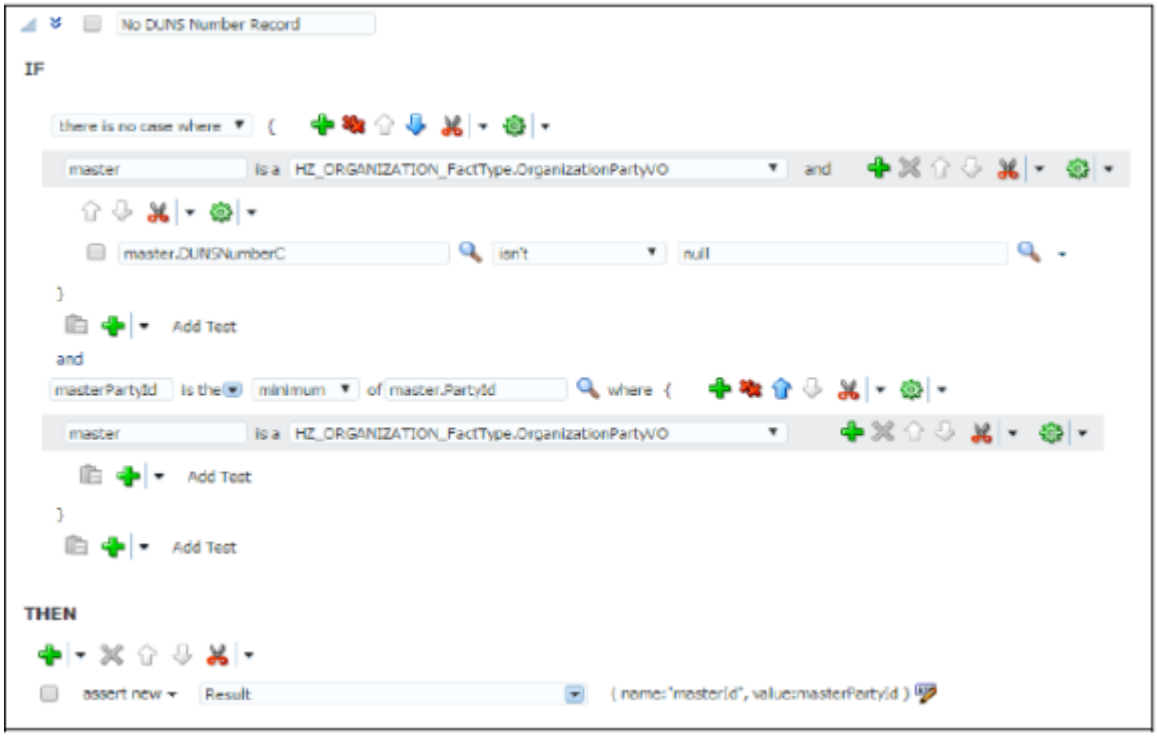
In this case, there is no record with D-U-N-S number. Therefore, the record with smallest party ID is picked as the master record. The following table lists the sample record information for this use case.

Record Name	Party ID	D-U-N-S Number	Master
Record 1	21	null	Yes
Record 2	22	null	No
Record 3	23	null	No

The following table lists the IF and THEN rules condition values that you must enter on the Define Survivorship Rules: Select Master Record page for this use case.

Rule Condition	Value
IF Condition	**Pick D-U-N-S number** **{IF master is an HZ_PERSON_FactType.OrganizationPartyVO** **master.DUNsNumberC isn't null}** **and** **masterPartyID is the minimum of masterPartyID** **where** **{master= HZ_PERSON_FactType.OrganizationPartyVO and** **master.DUNsNumberC isn't null}**
THEN Condition	**THEN** **Assert new Result (name:"masterId", value:" master.PartyId)**

The following figure displays the Define Survivorship Rules: Select Master Record page with completely filled IF and THEN rules conditions to set the record that has the smallest party ID as the master when no record with D-U-N-S number is found.



In this example, you have created two set master rules for Organization. First rule is for the cases where the input records have at least one record with D-U-N-S number. The second is for the case where no records have D-U-N-S Number.

> **Note:** You can activate more than one survivorship rule at a time. When you activate multiple rules, make sure that the rules aren't conflicting and the conditions in the rule are set according to the priority.

How You Set the Record with D-U-N-S Number and Highest Number of Party Sites as Master

This rule identifies and returns the master record based on the following three conditions in the order of priority listed:

1. Pick master that has D-U-N-S Number.
2. Pick master that has more party sites.
3. Pick master that has the smallest Party ID.

The following are two use cases for creating a set master rule to select the master record based on D-U-N-S number, number of party sites, and party ID:

Use Case 1

In this case, there are three records with D-U-N-S number and two records with highest number of party sites. Among those two records, the one with the lower value for party ID is selected as master. The following table contains the sample record information for this use case.

Record Name	Party ID	Number of Party Sites	D-U-N-S Number	Master
Record 1	11	3	198837472	Yes
Record 2	12	3	489203901	No
Record 3	13	2	384792392	No
Record 4	14	1	null	No

Use Case 2

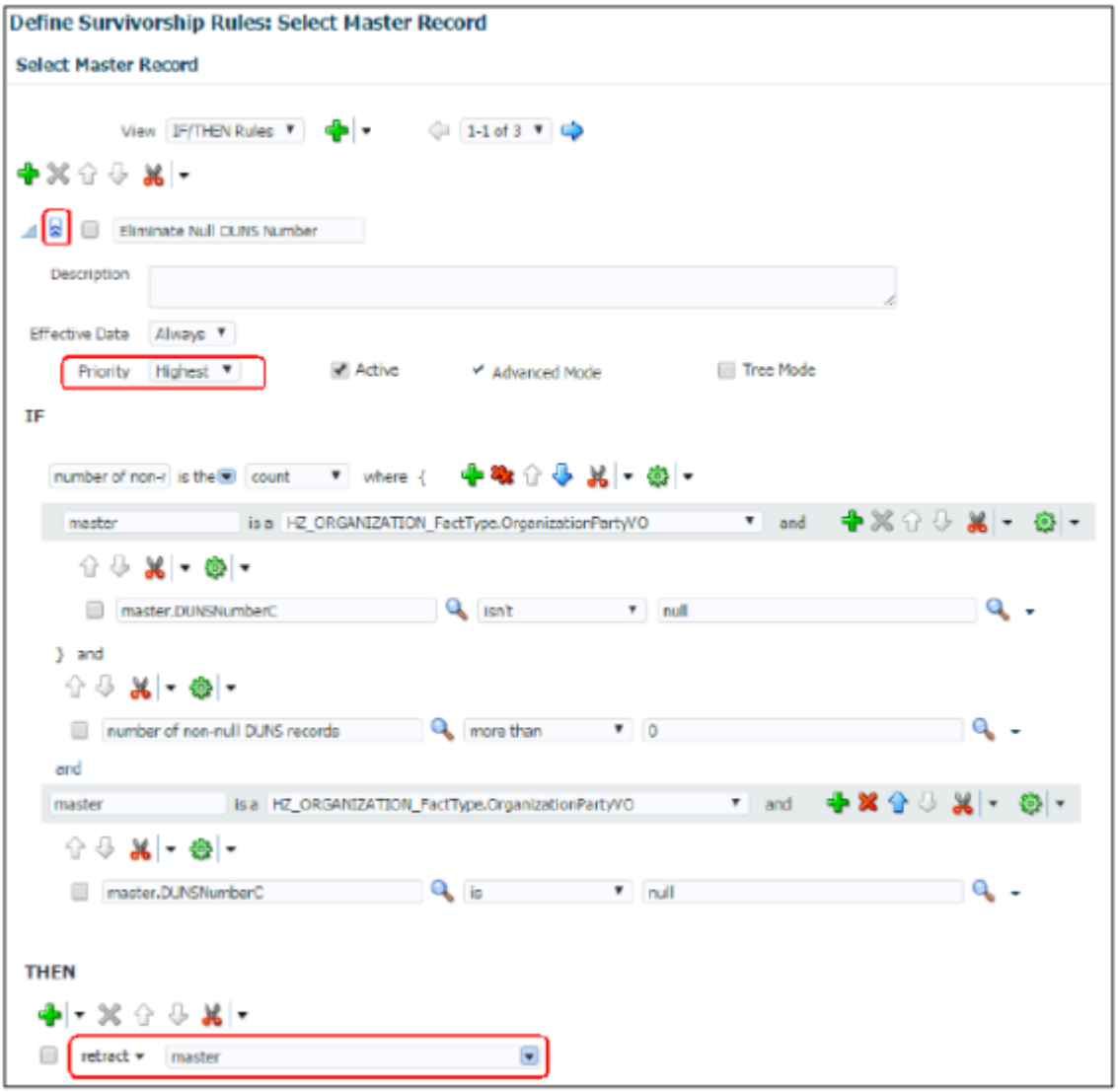
In this case, there are no records with D-U-N-S number. So, among the two records with higher number of party sites, the record with the smaller party ID is picked as the master record. The following table contains the sample record information for this use case.

Record Name	Party ID	Number of Party Sites	D-U-N-S Number	Master
Record 1	21	1	null	No
Record 2	22	2	null	Yes
Record 3	23	3	null	No

To create a master record with D-U-N-S number, number of party sites, and party ID, you add one more condition to the previous example where you set a master record with D-U-N-S number. Adding a condition to the previous example makes the rules complicated and cumbersome. Instead, you can create a simple rule for each condition to narrow down the list of potential master records and create another simple rule in the end to pick one record from the remaining potential master records.

Note: You can activate more than one survivorship rule at a time. When you activate multiple rules, make sure that the rules aren't conflicting and the conditions in the rule are set according to the priority.

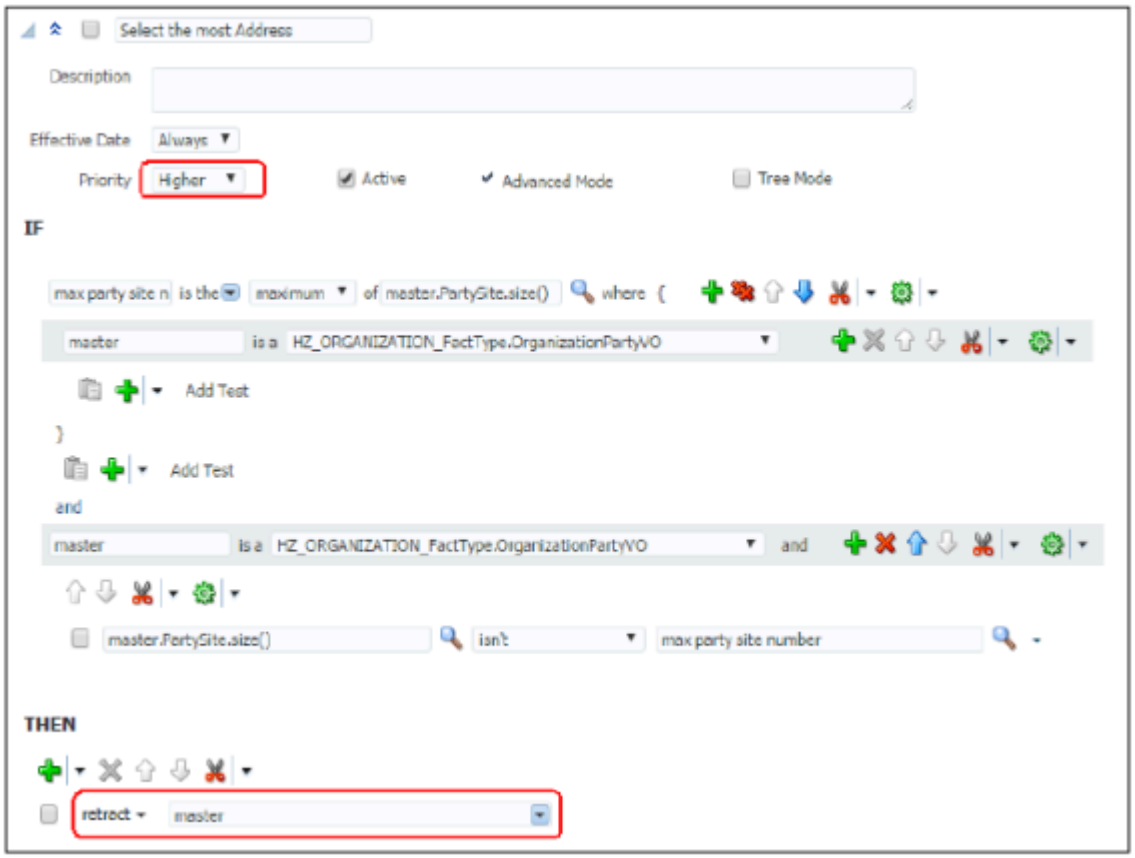
The following figure displays the Survivorship Rules: Select Master Record page with the IF and THEN rules condition values for creating a set master rule to set the record that has D-U-N-S Number as master. The priority of the rule is set as highest. The details of the conditions are as follows:Priority: HighestIF condition: If number of non-null DUNS records is the count where {master is a HZ_ORGANIZATION_FactType.OrganizationPartyVO and master.DUNSNumberC isn't null} and number of non-null DUNS records more than 0 and master is a HZ_ORGANIZATION_FactType.OrganizationPartyVO and master.DUNSNumberC is nullTHEN condition: Then retract master



Now, you set the conditions to set the record with the maximum number of party sites as the master.

The following figure shows the Survivorship Rules: Select Master Record page with the IF and THEN condition values to set the record with the maximum number of party sites as the master from the remaining potential records. The priority of the rule is set as higher. The details of the conditions are as follows.Priority: HigherIF condition: If maximum party site number is the maximum of master.PartySite.size() where {master is a HZ_ORGANIZATION_FactType.OrganizationPartyVO} and master is a HZ_ORGANIZATION_FactType.OrganizationPartyVO and master.PartySite.size() isn't maximum party site numberTHEN

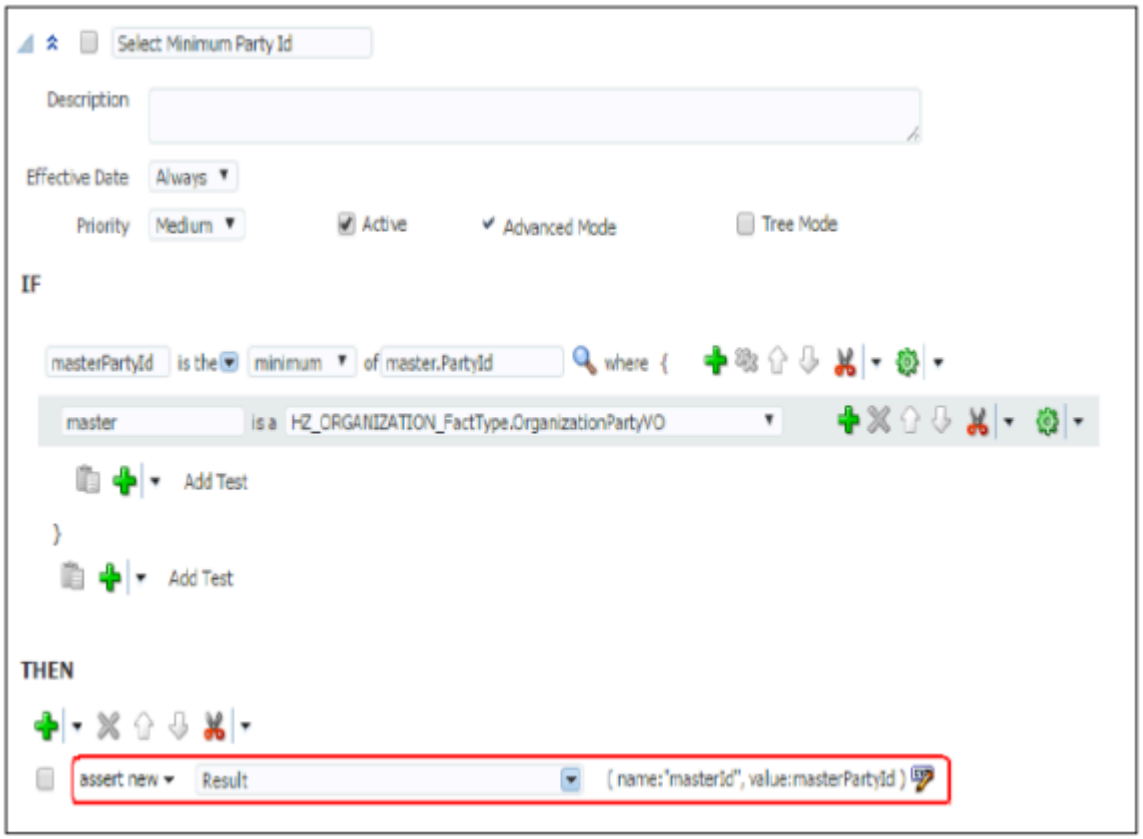
condition: Then retract master



When the records are screened with two previous conditions, you create a third condition to screen all remaining potential records with the smallest party ID.

The following figure shows the Survivorship Rules: Select Master Record page with the IF and THEN condition values for creating a set master rule to set the record with the smallest party ID as the master. The priority of the rule is set as medium. The details of the conditions are as follows.Priority: MediumIF condition: If masterPartyId is the minimum of master.PartyId where {master is a HZ_ORGANIZATION_FactType.OrganizationPartyVO}THEN condition: Assert new Result(name:

"masterId",value:masterPartyId)



In this example, the rules are created to narrow down the list of potential master records. When you activate more than one rule at a time you should set the conditions for the rule according to the priority to narrow down the list of potential master records. In this example Eliminate Null D-U-N-S Number rule is executed first to select records with D-U-N-S Number. Select the Most Address rule is executed next to find the master record with most number of party sites among the potential master records having D-U-N-S Number. Finally, Select Minimum Party ID rule is executed at the end to pick the minimum party ID from the remaining party records.

How do I define set attribute value rules?

This example demonstrates how to create a survivorship rule of the type Set Attribute Value.

You can determine survivorship at the attribute level using the set attribute value survivorship rule type. Set attribute value rules are used in party merge to determine which attribute value should come from which record.

The input to the survivorship rule is given in the IF clause. In a set attribute value rule, the inputs are the party records and their source information. The source information contains information about all attributes for each record in the database. If you're creating rules that use the Source information VO, you define it in the Define Source Systems Confidence page of the Manage Source System Confidence task. You must map each attribute to its source system and given a Source Confidence score on a scale of 1 to 100.

The following table lists the attributes in the source information VO to create a set attribute value rule.

Attribute Name	Description
RecordId	The record ID of a specific attribute.

Attribute Name	Description
AttributeName	The name of the attribute.
Source	The source system from where the attribute is updated.
SourceConfidenceLevel	The source confidence level assigned to the source system.
SourceUpdateDate	The date when the attribute was last updated.

The THEN clause determines the output object that picks the master record. In this case, setAttribute function creates the output object. To create Set Attribute Value rules, you perform the following steps:

1. Navigate to the Manage Survivorship Rules task.
2. Click Add. The Create Survivorship Rule page appears.

   Tip: You can select attributes from the available attributes in the Create Survivorship Rule page. It pre-populates the rule template with the selected attributes. It's not mandatory to set attributes from the available attribute.

3. Enter the sample information provided in the following table on the Create Survivorship Rule page.

Field	Value
Rule Name	PickAttributeValueRule
Description	Select the master value for a specific attribute based on specified master selection criteria.
Select the master value for a specific attribute based on specified master selection criteria.	Set attribute value
Object Type	Organization
Template	Select the Attribute Based template to select the surviving value based on the characteristic of the attributes. For example, you need an Attribute Based template to pick an attribute with the highest or lowest value such as a party number, or salary, or the earliest incorporated year. Select the Source Confidence Based template to select the surviving value based on the confidence in the source information.

4. Select attributes from the available attributes to pre-populate the rules template. In case you want to use the predefined set attribute rules, don't select any attributes.
5. Click Apply. You're taken to the Define Survivorship Rules: Select Attribute Value page.

In the Define Survivorship Rules: Select Attribute Value page, you specify criteria for selecting the master attribute value. The criteria that you define in this page determine the feature of the rule. You also have the option of using one of the following three predefined templates:

- Highest Source Confidence Level Wins for Organization (or Person): Use this rule to select the attribute values with the highest source confidence level.
- Most recently updated Organization (or person) attribute: Use this rule to select the attribute values with the most recently updated date.
- Least recently updated Organization (or person) attribute: Use this rule to select the attribute values with the oldest updated date.

The following sections of this topic contain three worked examples that show different ways of manually setting master attribute values in the Define Survivorship Rules: Select Attribute Value page. They are as follows:

- Set the Values with the Earliest Update Date as the Surviving Attribute Values
- Set the Value with the Highest Source Confidence Level as the Surviving Attribute Value for D-U-N-S Number
- Set the Values with the Earliest Incorporated Year as the Surviving Attribute Values

How You Set the Values with the Earliest Update Date as Surviving Attribute Values

This rule has a single condition to set all the surviving attribute values based on the earliest update date. The following is a use case for a set attribute rule to select the values with the earliest update date as surviving attribute values:

Use Case 1

Party Record

The following table contains information for party records.

Record Name	Party ID	Party Name	D-U-N-S Number
Record 1	1	Oracle Corp	198837472
Record 2	2	Oracle USA Corp	489203901
Record 3	3	Oracle	null

Source Information

The following table contains information for source information records.

Record ID	Attribute Name	Source	Source Confidence Level	Source Update Date
1	Party Name	FUSION	95	1/5/2016
2	Party Name	FUSION	95	1/5/2010
3	Party Name	SIEBEL	90	1/5/2000
1	D-U-N-S Number	DNB	100	2/5/1990

Record ID	Attribute Name	Source	Source Confidence Level	Source Update Date
2	D-U-N-S Number	FUSION	95	1/5/2016

In this case:

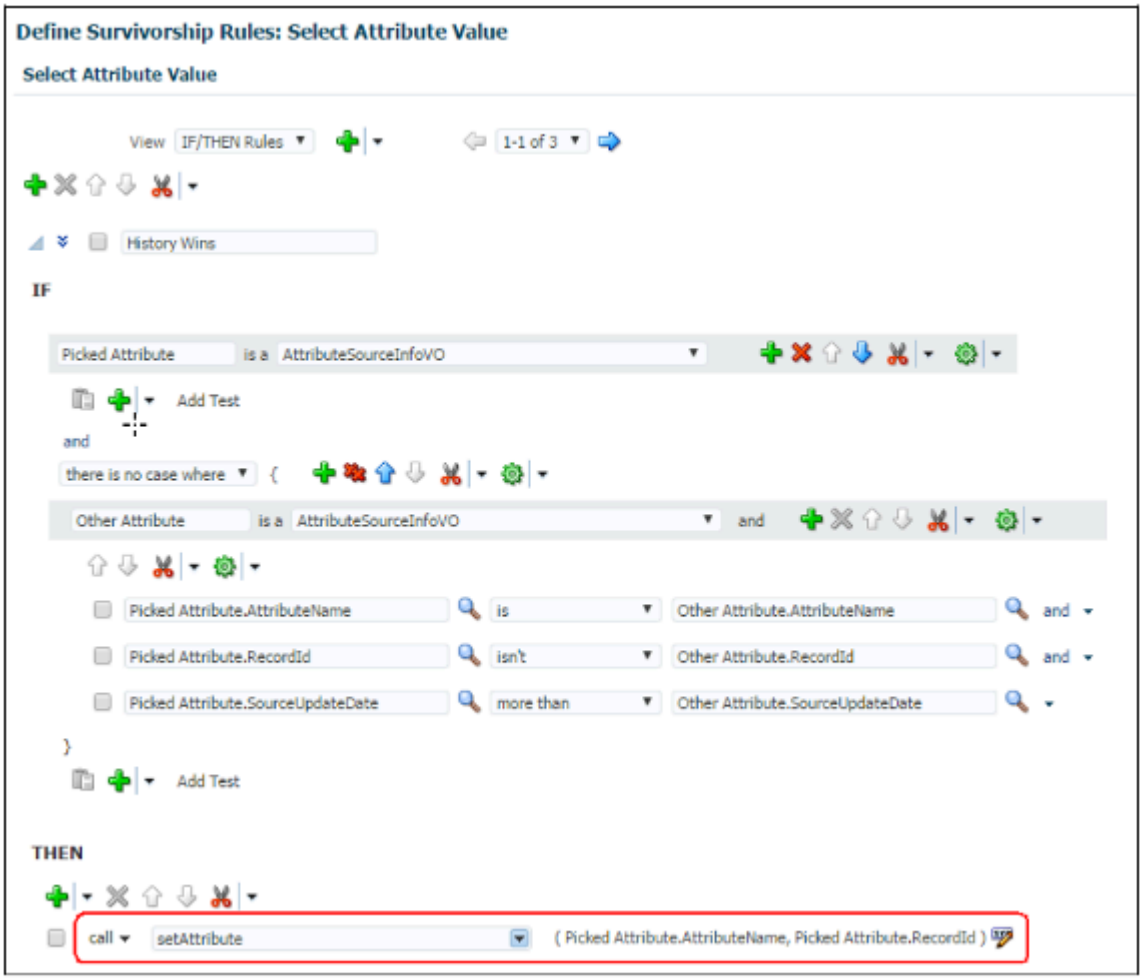
- The D-U-N-S number attribute value from the record with ID 1 is selected as master because the source information indicates that it has the earliest source update date.
- The Party Name attribute value from the record with the ID 3 is selected as master because the source information indicates that it has the earliest source update date.

Populate the Define Survivorship Rules: Select Attribute Values page with the IF and THEN rules condition values provided in the following table.

Rule Condition	Value
IF Condition	`IF picked attribute is a AttributeSourceInfoVO and there is no case where { Other Attribute == AttributeSourceInfoVO and there is no case where Other Attribute is a AttributeSourceInfoVO Picked Attribute.Attributename is OtherAttributename and PickedAttribute.RecordId isn't Other Attribute.recordId and Picked Attribute.SourceUpdateDate more than OtherAttribute.SourceUpdateDate}`
THEN Condition	`THEN call setAttribute (picked attribute.Attributename, Picked Attribute.RecordId)`

The following figure displays the Define Survivorship Rules: Select Attribute Values page with the IF and THEN rules conditions to create the set attribute value rule that selects the values with the earliest source update date as the surviving attribute value. The figure provides the following details. Name of the rule: History WinsIF condition: If picked attribute is a AttributeSourceInfoVO and there is no case where {Other Attribute is AttributeSourceInfoVO and Picked Attribute.AttributeName is Other Attribute.AttributeName and PickedAttribute.RecordId isn't Other Attribute.RecordId and Picked Attribute.SourceUpdateDate more than Other

Attribute.SourceUpdateDate}THEN condition: Then call setAttribute (Picked Attribute.AttributeName, Picked Attribute.RecordId)



How You Set the Value with the Highest Source System Confidence Level as the Surviving Attribute Value for D-U-N-S Number

This rule has a single condition to select the D-U-N-S number value with the highest source confidence level as the surviving attribute value for the D-U-N-S number attribute. The following is a use case for a set attribute rule to select the D-U-N-S number value with the highest source confidence level as the surviving attribute value:

Use Case

Party Record

The following table contains information for party records.

Record Name	Party ID	Party Name	D-U-N-S Number
Record 1	1	Oracle Corp	198837472
Record 2	2	Oracle USA Corp	489203901
Record 3	3	Oracle	null

Record Name	Party ID	Party Name	D-U-N-S Number

In this case, the party record contains three records with the attribute Party Name and two records with D-U-N-S number. These attributes are picked to create the source information. The source information table defined using the attributes from this party record table is as follows:

Source Information

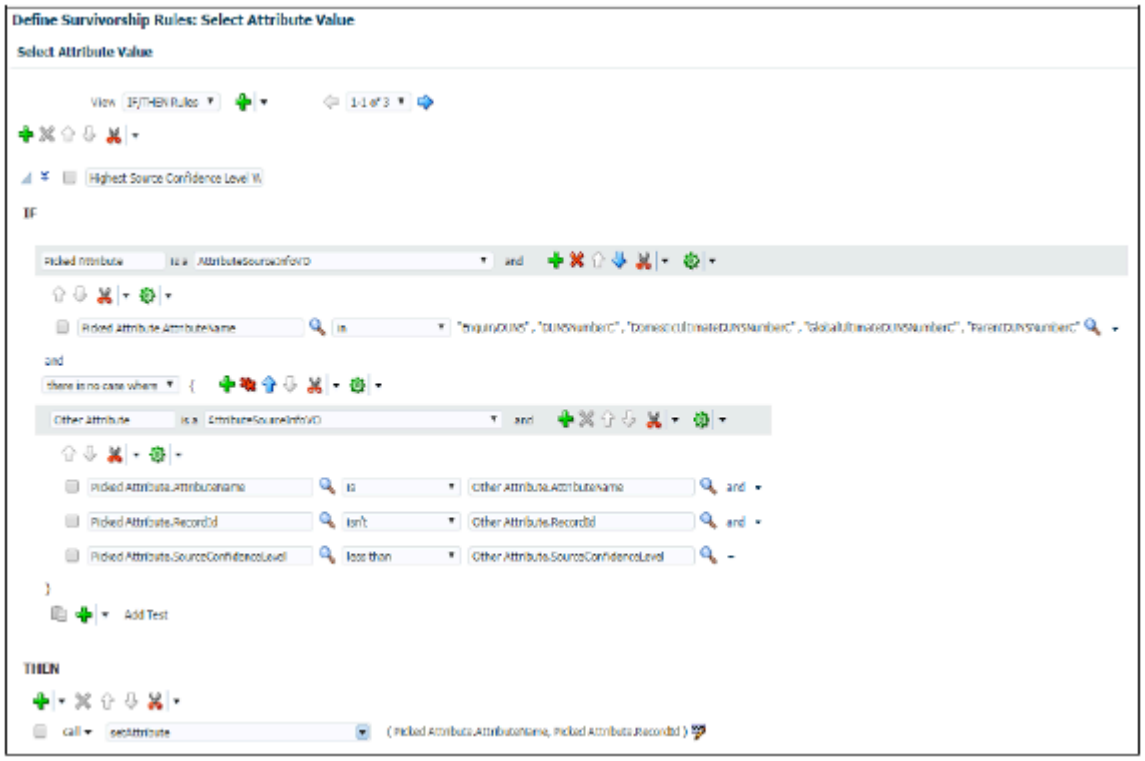
The following table contains information for source information records.

Record ID	Attribute Name	Source	Source Confidence Level	Source Update Date
1	Party Name	FUSION	95	1/5/2016
2	Party Name	FUSION	95	1/5/2010
3	Party Name	SIEBEL	90	1/5/2000
1	D-U-N-S Number	DNB	100	2/5/1990
2	D-U-N-S Number	FUSION	95	1/5/2016

In this case, the D-U-N-S attribute value for the record with ID 1 is selected as master because the source information indicates that it has the highest source confidence level among all records that have the D-U-N-S Number attribute.

The following figure displays Define Survivorship Rules: Select Attribute Values page with the IF and THEN rules condition values to set the attribute value with the highest source system confidence level as the master. The figure provides the following details.Name of the rule: Highest Source Confidence LevelIF condition: If picked attribute is a AttributeSourceInfoVO andPicked Attribute.ArrtibuteName is EnquireDUNSNumberC and there is no case where {Other Attribute is a AttributeSourceInfoVO and Picked Attribute.AttributeName is Other Attribute.AttributeName and Picked Attribute.RecordId isn't Other Attribute.RecordId and Picked Attribute.SourceConfidenceLevel is less than

OtherAttribute.SourceConfidenceLevel}THEN condition: Then call setAttribute (Picked attribute.AttributeName, Picked Attribute.RecordId)



How You Set the Values with the Earliest Incorporated Year as the Surviving Attribute Values

This rule has a single condition to select values with the earliest incorporated year as the surviving attribute values. The following is a use case for creating such a set attribute value rule:

Use Case 1

Party Record

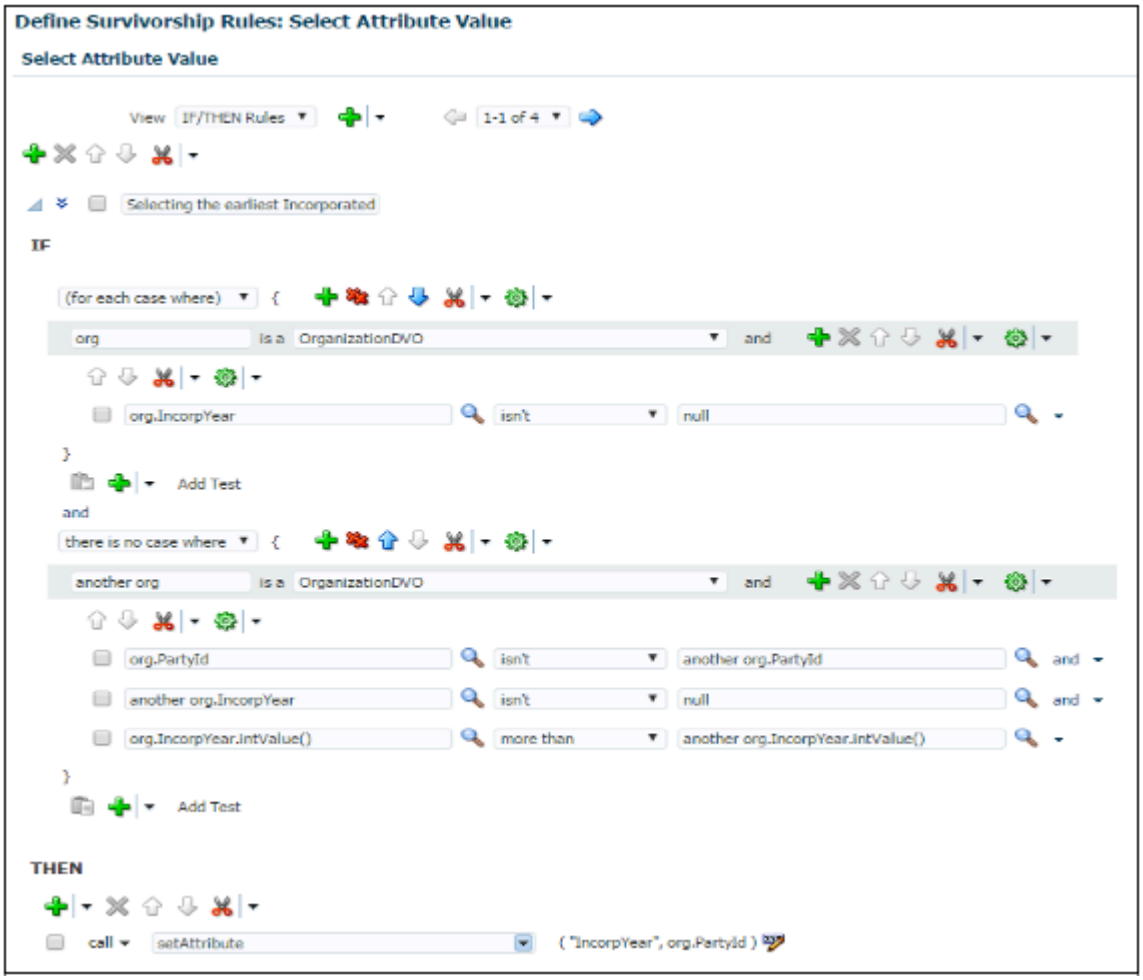
The following table contains information for party records.

Record Name	Party ID	Party Name	Incorporated Year
Record 1	1	Oracle Corp	1980
Record 2	2	Oracle USA Corp	1990
Record 3	3	Oracle	2000

In this case, the party record table contains three records with the attributes Party Name, Party ID, and Incorporated Year. The attribute values for the record with the earliest incorporated year are picked as master attribute values.

The following figure displays Define Survivorship Rules: Select Attribute Values page with the IF and THEN rules condition values to set the attribute with the earliest incorporated year as the master. The figure provides the following details.Name of the rule: Selecting the Earliest Incorporated YearIF condition: If for each case where org is a OrganizationDVO and org.IncorpYear isn't null and there is no case where another org is a OrganizationDVO and org.PartyId isn't another org.PartyId and another org.IncorpYear isn't null and org.IncorpYear.intvalue() is

more than another org.incorpyear.intValue()THEN condition: Then call setAttribute ("IncorpYear", org.PartyId)



What are the survivorship rules for parties with custom objects and attributes?

This topic describes how to define survivorship rules for merging parties having custom objects and attributes.

For example, you can define a survivorship rule of the type Set Master Record to select the party with the maximum number of custom objects or the party with a specific attribute as the master. You can also define a survivorship rule of the type Set Attribute Value to select the smallest value for a specific attribute.

Custom Objects and Attributes

You can use Oracle Application Composer to configure and extend Oracle CX Sales and Fusion Service applications. The Application composer provides you the ability to extend an Oracle CX Sales and Fusion Service application's object model. You can configure CX Sales and Fusion Service objects by adding new fields (custom fields or custom attributes) to an existing object (standard objects). Or you create entirely new objects (custom objects) and related fields (custom attributes).

Custom Objects and Attributes in Survivorship Rules

The following table describes the basic merge operations that can be performed on standard or custom objects.

Object	Examples	Can be merged	Can be transferred	Can be removed
Standard Top-Level Object	Account/Organization Contact/Person	Yes	Not applicable for top-level objects	Not applicable for top-level objects
Standard Child Object	Address Phone	Yes (by reviewing in the Data Steward UI)	Yes (by default if not merged or removed)	Yes (by reviewing in the Data Steward UI)
Standard Child Object	Lead Opportunity	No	Yes (always)	No
Custom Top-Level Object	Not applicable (implementation specific)	No	Not applicable for top-level objects	Not applicable for top-level object
Custom Child Object	Not applicable (implementation specific)	No	Yes (always)	No

The following table describes whether or not these objects and attributes can be used while defining conditions, clauses, and actions for survivorship rules of the type Set Master Record and Set Attribute Value.

Object	Examples	Can be used in Set Master Record rule conditions	Can be used in Set Master Record rule actions	Can be used in Set Attribute Value rule conditions	Can be used in Set Attribute Value rule actions	Can be used for Attribute Override in Data Steward UI
Standard Top-Level Object	Account/ Organization Contact/Person	Yes (standard and custom attributes)	Yes (standard record ID attribute)	Yes (standard and custom attributes)	Yes (standard and custom attributes)	Yes (standard and custom attributes)
Standard Child Object	Address Phone	Yes (standard and custom attributes)	No, survivorship not supported for child objects	No, survivorship not supported for child objects	No, survivorship not supported for child objects	No, attribute override not supported for child objects
Standard Child Object	Lead Opportunity	Yes (standard and custom attributes)	Not applicable if you can't merge the records	No	Not applicable if you can't merge the records	Not applicable if you can't merge the records
Custom Top-Level Object	Not applicable (implementation specific)	No	Not applicable if you can't merge the records	No	Not applicable if you can't merge the records	Not applicable if you can't merge the records
Custom Child Object	Not applicable (implementation specific)	Yes (all attributes are custom)	Not applicable if you can't merge the records	No	Not applicable if you can't merge the records	Not applicable if you can't merge the records

Example of Defining Set Master Record Rules for Parties with Custom Attributes

This topic explains how to define a survivorship rule of the type Set Master Record to select the party with a specific custom attribute as the master.

This rule identifies and returns the master record based on the following two conditions:

1. Pick master that has a specified custom attribute. In this case the custom attribute is Testfieldone_c.
2. Pick master that has the highest Party ID.

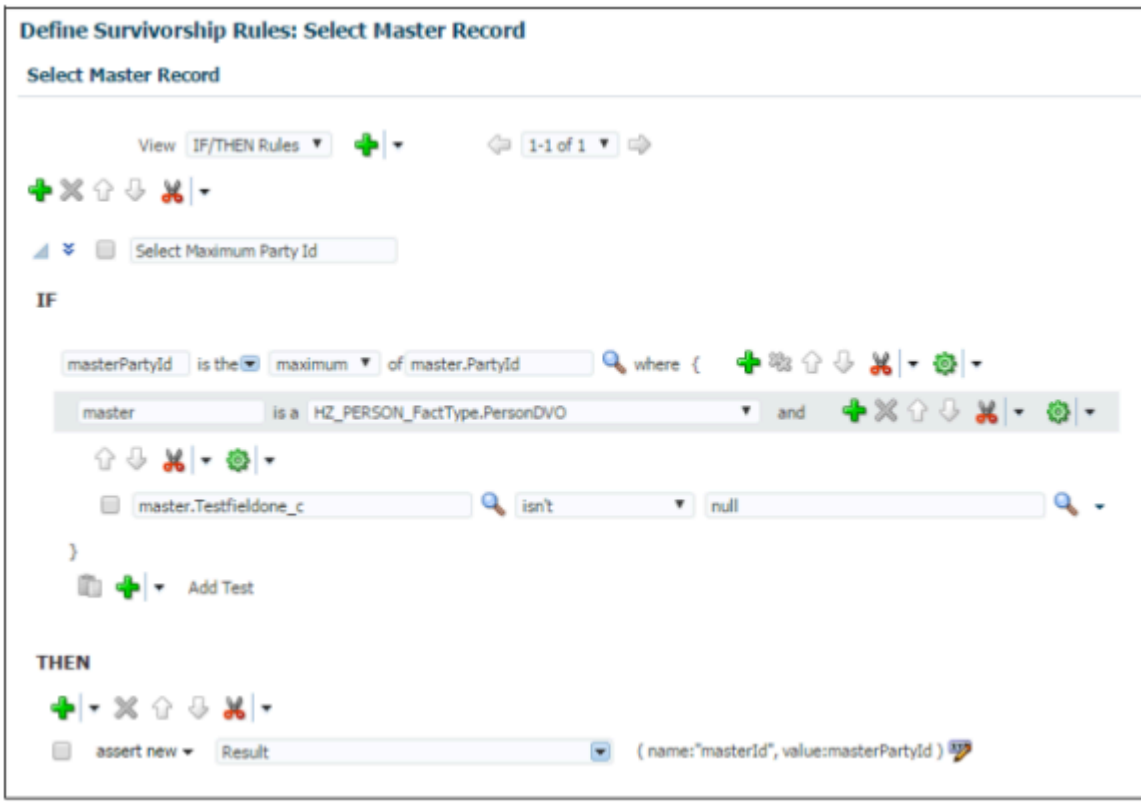
The following table provides the details of a use case for setting master record rule to select the party with a specific custom attribute as the master.

Record Number	Party ID	Testfieldone_c	Master
Record 1	11	Test	No
Record 2	12	Test	Yes
Record 3	13	None	No
Record 4	14	None	No

In this case, there are two records with the custom attribute Testfieldone_c. Therefore, the record with highest party ID is picked as the master record.

The following figure displays Define Survivorship Rules: Select Master Record page with the IF and THEN rules condition values to create the set master record rule to pick the record with specified attributes as the master. The details of the conditions are as follows:IF condition: If masterPartyId is the maximum of master.PartyId where {master is a HZ_PERSON_FactType.PersonDVO and

master.Testfielddone_c isn't null}THEN condition: Then Assert new Result (name:"masterId", value:masterPartyId)



Example of Defining Set Master Record Rules for Parties with Custom Attributes

This rule identifies and returns the master record based on the following two conditions:

- Pick master that has a specified custom attribute. In this case the custom attribute is Testfieldone_c.
- Pick master that has the highest Party ID.

The following table provides the details of a use case for setting master record rule to select the party with a specific custom attribute as the master.

Record Number	Party ID	Testfieldone_c	Master
Record 1	11	Test	No
Record 2	12	Test	Yes
Record 3	13	None	No
Record 4	14	None	No

In this case, there are two records with the custom attribute Testfieldone_c. Therefore, the record with highest party ID is picked as the master record.

The following figure displays Define Survivorship Rules: Select Master Record page with the IF and THEN rules condition values to create the set master record rule to pick the record with specified attributes as the master. The details of the conditions are as follows:

IF condition: If masterPartyId is the maximum of master.PartyId where {master is a HZ_PERSON_FactType.PersonDVO and master.Testfielddone_c isn't null}

THEN condition: Then Assert new Result (name:"masterId", value:masterPartyId)Define Survivorship Rules: Select Master Record This image depicts defining survivorship.

Example of Defining Set Attribute Value Rules for Custom Attributes

This topic explains how to define a survivorship rule of the type Set Attribute Value to select the smallest value for a specific attribute. This rule has a single condition where it picks a record that has the smallest value for the specified custom attribute.

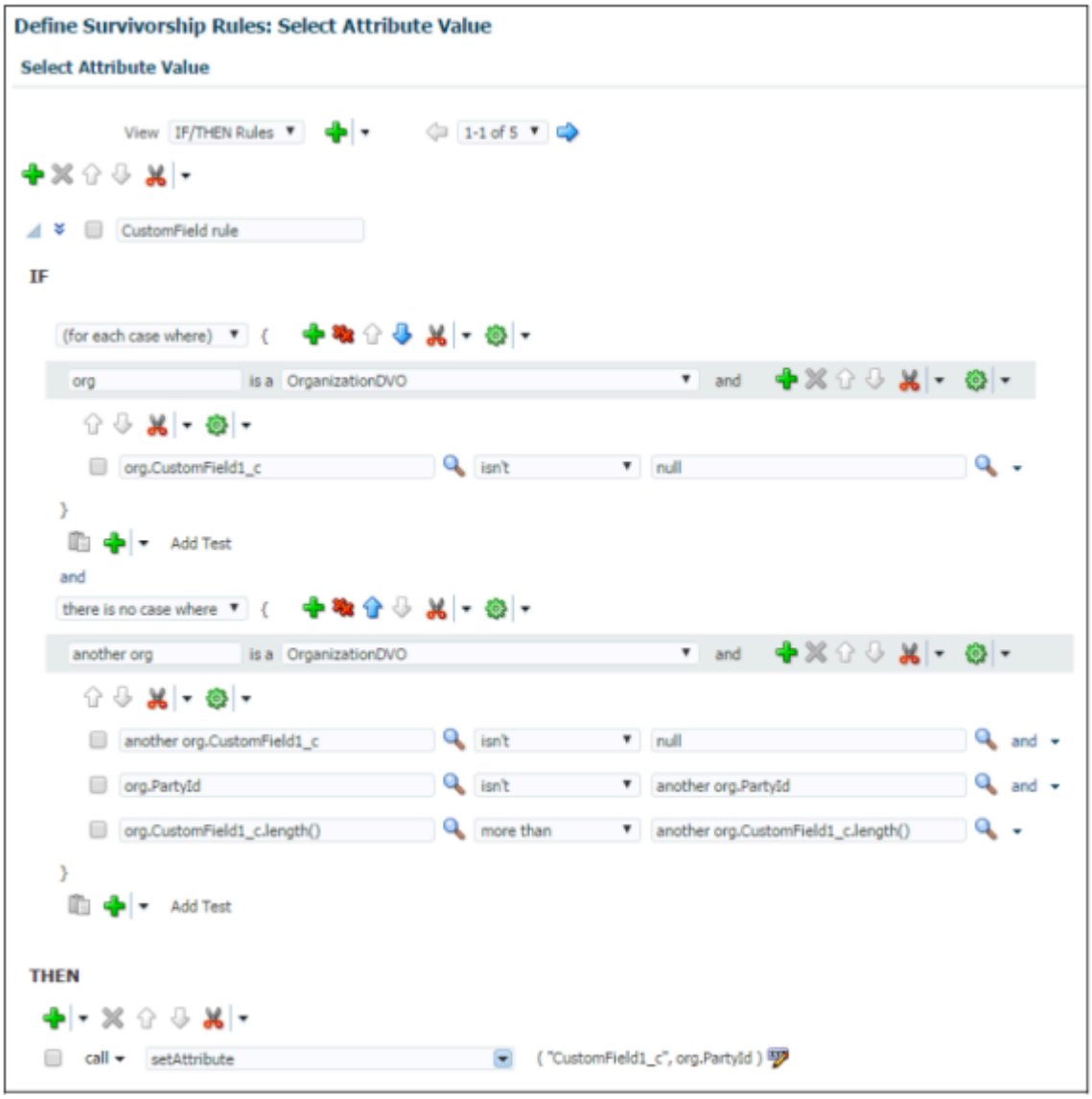
The following table contains the details of a use case for a set attribute value rule to select the smallest value for a specific custom attribute as the survivor. In this case the example attribute name is CustomField1_c.

Record Number	Party ID	CustomField1_c Attribute Value	Party Name	Survivor
Record 1	11	123	Oracle Corp	Yes
Record 2	12	1456	Oracle USA Corp	No
Record 3	13	239940	Oracle	No

In this case, there are three records with CustomField1_c custom attribute. The value 123 is picked up as the survivor value for the CustomField1_c custom attribute because it's the smallest value for that attribute.

The following figure displays the Define Survivorship Rules: Select Attribute Value page with the IF and THEN rules condition values to create the set attribute value rule to pick the smallest value for the CustomField1_c custom attribute as the survivor. The details of the conditions are as follows:IF condition: If (for each case where) {org is a OrganizationDVO and org.CustomField1_c isn't null } and there is no case where {another org is a OrganizationDVO and another org.CustomField1_c isn't null and org.PartyId isn't another org.PartyId and org.CustomField1_c.length()

more than another org.CustomField1_c.length()}THEN condition: Then call setAttribute("CustomField1_c", org.PartyId)



How can I check if the survivorship rules that I created work?

You can create two parties and merge them into a single record to check if your survivorship rules are working. You must ensure that your survivorship rules are active before merging the parties.

What are the seeded sample survivorship rules that Oracle Fusion Applications provide by default?

Oracle Fusion Applications offers some sample survivorship rules.

Here are predefined sample survivorship rules:

- For organizations, history wins: The attribute value with earlier source update date wins. This rule is applicable for all organization attributes.
- For persons, recent wins: The attribute value with later source update date wins. This rule is applicable for all person attributes

- For organizations, assign the highest source confidence level to the D-U-N-S number.

The seeded survivorship rules are in inactive status out-of-the-box. These rules are updated with every release. However, once you change these rules, these rules aren't updated.

What's the difference between survivorship rules and merge agreement rules?

Survivorship rules are a collection of business rules defined for the creation of the best version record intelligently, especially from multiple source systems, by specifying criteria for selecting record and attributes to be retained during merge or link operations.

A merge agreement rule is a collection of patterns and conditions that are defined to determine whether a merge request should be vetoed by the application or not. Merge requests that violate these rules are either rejected or end in error.

Agreement Rules

What are Agreement Rules?

An agreement rule is a collection of patterns and conditions that are defined to determine whether a merge request should be vetoed by the application or not. Merge requests that violate these rules are either rejected or end in error.

Agreement rules let you check a merge request for any veto conditions that can prevent a merge from occurring. These rules save resources and time by obviating the need to review merge requests to prevent undesired merge from being processed. Besides, agreement rules prompt you to consider alternative duplicate resolution mechanism such as linking.

Agreement rule can be of the following two types:

- Predefined
- User-defined

Predefined Agreement Rules

These are agreement rules that are predefined in the application. You can only view the predefined agreement rules.

The following table describes the predefined agreement rules shipped with the application for Oracle Sales Cloud. Merge requests that violate these rules are automatically rejected.

Name	Description
`HR_APPLICANT_VETO`	Prevents a person party with an active party usage of `HR_APPLICANT` from merging with another party.
`HR_EMPLOYEE_VETO`	Prevents a person party with an active party usage of `HR_EMPLOYEE` from merging with another party.
`HR_CONTINGENT_WORKER_VETO`	Prevents a person party with an active party usage of `HR_CONTINGENT_WORKER` from merging with another party.
`HR_PARTY_SITE_VETO`	Prevents a party site with active original system reference from Oracle Fusion Human Capital Management system from merging with another party site.
`RESOURCE_PERSON_VETO`	Prevents a person party with an active party usage of `RESOURCE` from merging with another party.

Name	Description
BANK_VETO	Prevents an organization party with an active party usage of BANK from merging with another party.
CLEARINGHOUSE_VETO	Prevents an organization party with an active party usage of CLEARINGHOUSE from merging with another party.
BANK_BRANCH_VETO	Prevents an organization party with an active party usage of BANK_BRANCH from merging with another party.
BRANCH_CLEARINGHOUSE_VETO	Prevents an organization party with an active party usage of CLEARINGHOUSE_BRANCH from merging with another party.
LEGAL_EXTLEGAL_PARTYUSGRULE_VETO	Prevents an organization party with active party usage of LEGAL_ENTITY from merging with another organization party with active party usage of EXTERNAL_LEGAL_ENTITY.
IC_PARTICIPANT_PERSON_VETO	Prevents a person party with an active party usage of INCENTIVE_COMP_PARTICIPANT from merging with another party.
IC_PARTICIPANT_ORG_VETO	Prevents an organization party with an active party usage of INCENTIVE_COMP_PARTICIPANT from merging with another party.
PARTNER_VETO	Prevents an organization party with an active party usage of PARTNER from merging with another party.
INACTIVE_PARTNER_VETO	Prevents an organization party with an active party usage of INACTIVE_PARTNER from merging with another party.
CUST_CONTACT_DIFF_RESOURCE_ORG_VETO	Prevents two partner owned contacts belonging to different resource organizations from being merged.
CUST_CONTACT_INTERNAL_PARTNER_VETO	Prevents a partner owned contact that doesn't belong to a resource organization from merging with another partner owned contact that belongs to a resource organization.
CUST_CONTACT_INTERNAL_PARTNER_ORG_VETO	Prevents a contact owned by an internal user from merging with a contact owned by a partner user.
CARD_ISSUER_VETO	Prevents an organization party with an active party usage of CREDIT_CARD_PROVIDER from merging with another party.
LEGAL_ENTITY_VETO	Prevents an organization party with an active party usage of LEGAL_ENTITY from merging with another party.
ESTABLISHMENT_VETO	Prevents an organization party with an active party usage of ESTABLISHMENT from merging with another party.
HIERARCHY_CYCLE_PREVENTATION_VETO	Prevents merge from happening when the master party record is at a level lower than the nonmaster party record within the same active tree version.

Name	Description
`NAMEDACCOUNT_UNNAMEDACCOUNT_VETO`	Prevents merge from happening when the master party record isn't a named account whereas the `nonmaster party record`is a named account and these are in active hierarchies.

The following table describes the predefined agreement rules shipped with the application for other Fusion applications. Merge requests that violate these rules are automatically rejected.

Functional Area	Name	Description
Distributed Order Orchestration	DOOPartyMerge	Prevents the customer account and customer account sites from getting merged when an open Distributed Order Orchestration (DOO) order is associated with either of them.
Distributed Order Orchestration	POZSUPPLIER	Prevents the party merge process from being completed, when any of the following conditions are met: • The corresponding supplier isn't merged • Few purchase orders aren't transferred for the corresponding From Supplier • The merged-from party and merged-to party aren't correlated with the same merged-from supplier and merged-to supplier • The corresponding supplier site haven't been merged • Few purchase orders aren't transferred for the corresponding supplier site • The merged to party site doesn't have an associated supplier site • The merging contact parties aren't valid supplier contacts • The merging contact parties don't belong to same supplier
Tax	ZX_PARTY_MERGE	Prevents the party from merging when any of the following conditions are met: • The 'From Party Tax Profile type' is not the same as 'To Party Tax Profile type' • The 'From Party Tax Profile type' equal to 'THIRD PARTY' • The From party tax attributes (Registration Type code, Registration Number, Rule code, Self-assess flag, Inclusive tax flag) aren't same as the To party tax attributes (Registration Type code, Registration Number, Rule code, Self-assess flag, Inclusive tax flag)

Functional Area	Name	Description
Accounts Payable	AP_PARTYVETO	Prevents party merge in the unprocessed invoices or unprocessed payments.
Accounts Receivable	AR_FinArCustomersVeto	Prevents the customer merge when either of the following conditions are met: • The merging customer has any Balance Forward Bill (BFP) in draft mode in the application • The merging customer site has any BFB in draft mode in the receivables application
CSE Assets	CSE_SELLING_BUSINESS_UNIT_VETO	After the customer account site merge, CSE party merge will check if the new account site + the sold from business unit combination that we have in our cse_assets_b table is a valid combination using hz_cust_acct_sites_all. If not valid, we will veto the party merge.
Service	SVC_CSS_CONT_PENDING_REG_VETO SVC_CSS_ACT_PENDING_REG_VETO SVC_CSS_ACT_USERS_VETO	Validate that contacts can't be merged if the victim contact has a self-service registration record Don't allow account merge if the account has contacts with pending registrations Do not allow account merge if the account has active ODCS users
Grant/Award Management	GMS_SPONSOR_NOT_EXIST_VETO GMS_TOSPON_NOTLOC_VETO	The master party doesn't exist as a grants sponsor. The sponsor LOC of the master and target party isn't same. Ensure the LOC is same for both sponsors. Either the LOC for both is enabled or both is disabled.

User-defined Agreement Rules

These are additional agreement rules that you can define to determine whether a merge request should be vetoed by the application. You can create, view, update, and delete user-defined agreement rules.

> **Note:** The application doesn't support changing agreement rules inside the Application Composer sandbox. Therefore, the merge engine doesn't pickup the changes made to these rules inside the Application Composer sandbox.

Agreement Rules Dictionary

An agreements rules dictionary is a collection of predefined terms and attributes that can be used to define agreement rules.

The Oracle Customer Data Hub comes with a single predefined dictionary (HZ_Parties) that contains all the predefined Agreement Rules shipped out of the box with the application. You can also use this dictionary to define custom agreement rules according to your business requirements. Note that you can only view the predefined agreement rules. You can't edit them. In contrast, you can create, view, update, and delete user-defined agreement rules. Merge requests that violate agreement rules are rejected.

Before using this dictionary to define custom agreement rules, you must review whether the agreement rule terms and term attributes existing in the predefined agreement rules dictionary are sufficient to define custom agreement rules needed to meet your business requirements. If required, refresh terms to import the latest terms, term attributes, and related metadata, for example, fact types such as entities and objects. Refreshing the dictionary help you pull in all the newly added custom attributes for accounts and contacts.

Define Agreement Rules

This example demonstrates how to create user-defined agreement rules that you can use to prevent a merge request from being processed.

Agreement rules are collections of patterns and conditions that are defined to determine whether a merge request should be vetoed by the application or not. Perform the following tasks to define agreement rules:

- Review and refresh terms in the predefined agreement rules dictionary shipped out of the box
- Add a new agreement rule

For more information on agreement rules, see *Oracle Fusion Middleware User's Guide for Oracle Business Rules* on Oracle Technology Network at http://www.oracle.com/technetwork.

Review and Refresh Terms in the Predefined Agreement Rules Dictionary

The Customer Hub application is shipped with a predefined agreement rules dictionary that contains all the predefined Agreement Rules shipped out of the box with the application. Before using this dictionary to define custom agreement rules, you must review whether the agreement rule terms and term attributes existing in the predefined agreement rules dictionary are sufficient to define custom agreement rules needed to meet your business requirements. If required, refresh terms to import the latest terms, term attributes, and related metadata, for example, fact types such as entities and objects. Refreshing the dictionary helps you pull in all the newly added custom attributes for accounts and contacts. Use the following steps to review and refresh the agreement rules dictionary:

1. In the Setup and Maintenance work area, go to the following:
    - Offering: Customer Data Management
    - Functional Area: Customer Hub
    - Task: Manage Agreement Rules
2. On the Manage Agreement Rules page, review whether the agreement rule terms and term attributes existing in the predefined agreement rules dictionary are sufficient to define custom agreement rules needed to meet your business requirements.
3. Click **Refresh Terms** to import the latest terms, term attributes, and related metadata.
4. Click **OK** in response to the confirmation message.

Add a New Agreement Rule

After reviewing and refreshing the Agreement Rules Dictionary using the earlier steps, perform the following steps to create a new custom agreement rule:

1. On the Manage Agreement Rules page, click **Next** to navigate to the Manage Agreement Rules: Define Rules page.
2. Click **Add** form the Actions menu to add a new rule.
3. Enter a rule name.
4. Click **Define Rule**.
5. Enter the reason for creating the agreement rule in the **Justification Reason**.
6. Click **Add** form the Actions menu to create a new pattern.
7. Complete the fields in the new pattern field using the sample information provided in the following table. Use the default values except where indicated. Note that the relation is always AND between patterns and can't be edited. You must include the Dictionary Terms OrganizationPartyVO and PersonPartyVO, with defined MergeType, into the Define Patterns column. These patterns determines the master and nonmaster records.

Pattern	Dictionary Term	Term Alias	Relation
for each case where	PersonPartyVO	Person	AND
for each case where	OrganizationPartyVO	NonmasterParty	AND
there is a case where	PartyUsageAssignmentVO	PartyUsageAssignment	AND

8. Navigate to the Conditions table.
9. Click **Add** from the Actions menu to add a new condition and complete the fields using the sample information provided in the following table. Use the default values except where indicated.

Term Attribute	Operator	Value	Relation
Person.PartyNumber	is not	1234	AND
NonmasterParty.MergeType	=	Nonmaster	AND
UsageAssignment.PartyUsag	=	HR_APPLICANT	AND

10. Click **Save** or **Save and Close**.
11. Click **Submit**.

Source System Confidence

Source System Confidence Levels

Source system confidence levels indicate the reliability of a particular source system for specific attributes. They are used to determine the master or surviving record among multiple duplicate records from different source systems.

For each source system, you can set source confidence score for attributes in the Person and Organization objects. The scores are used to select attributes for the master record during merge operation. The scores are used to select attributes, based on the survivorship rules, for the master record during merge operation. You assign source confidence levels based on your understanding of the quality of the data stored in the various source systems within your organization.

Source system confidence levels range from 0 to 100.

How Source System Confidence Levels Work With Survivorship Rules

You can use source system confidence levels for creating survivorship rules. Survivorship rules are used to retain master records during merge, insert, or update operations.

Source system confidence levels

You can import data from legacy or source systems into the application. The legacy or source systems usually store data associated with the same attributes across systems. Similarly, multiple source systems may have data related to the same account.

Source system confidence levels let you indicate how much you trust the data from a specific source system for an attribute. For example, you may trust the customer address data coming from a financial system more than the same data from a marketing system. Similarly, you may trust the employee name data coming from the Human Relationships system more than the same data coming from the Marketing system. While specifying source system confidence levels, you must ensure that only the most reliable data is selected as the master data.

Survivorship rules

Oracle CX Sales and Fusion Service uses a central data model updated by authorized users from multiple applications. Often, different applications or source systems update the same data resulting in a data conflict.

Survivorship rules are custom business rules that determine how conflicts should be automatically resolved while merging or updating duplicate records from different source systems or applications. For example, you can create a survivorship rule to select the latest customer address data when there are duplicate records.

Relationship between source system confidence levels and survivorship rules

Source system confidence levels enable you to build your survivorship rules to resolve duplicate data associated with specific attributes. For example, you could create survivorship rules to:

- Select the record from a source system with confidence values of 80 and above as the master record.
- Automatically reject records from source systems with confidence values of 30 and below.

Synchronization of Attribute Change History for New Attributes

When a new attribute is configured for source confidence, you can synchronize the survivorship on update with that attribute's change history. Synchronization is relevant for the first-time a given attribute is configured with source confidence. Changes to an attribute's existing source confidence setup don't require synchronization. Synchronization isn't required to track changes later.

Here's how you synchronize an attribute's source history:

1. In the Setup and Maintenance work area, go to the following:
    - Offering: Customer Data Management
    - Functional Area: Customer Hub
    - Task: Manage Source System Confidence
2. Click Synchronize.

    The sync dialog provides information about the most recently run synchronization job.
3. Select the Object you want to synchronize on the Synchronize Attribute Source History.
4. You can select the **Recommended synchronization start date** option or select **Specify the synchronization start date** option and specify a date.
5. (Optional) Select **Send Notifications** to receive notifications when the synchronization job completes.
6. Click **Submit**.

> **Note:** The time taken for this process to complete depends on the number of records to be processed. If this process is taking a long time complete, you can view Last Sync Date and Sync Request ID columns and refresh the page to see the progress. You can also go to the Scheduled Processes page to check the status using the Sync Request ID.

FAQs for Source System Confidence

What are Set Master and Set Attribute rules?

Set Master rules enable you to define the logical business criteria for selecting the master record from a set of potential duplicate records for merge operations.

Set Attribute rules enable you to define the logical business criteria for selecting the best attribute values from multiple input records during merge operation.

Can multiple attributes of an object type have the same source system confidence level?

Yes, you can apply the same source system confidence level to multiple attributes of an object type. You can manage source system confidence levels using the following in the Setup and Maintenance work area: offering: Customer Data Management; functional area: Customer Hub; task: Manage Source System Confidence.

Automerge

The automerge functionality merges duplicate records without any approval or intervention from the data steward. Automatic processing of merge requests is critical when processing large volumes of customer data as automerge can expedite the resolution of duplicate records without manual review.

During automerge, the child entities of the duplicate records, such as contact points, relationships, classifications, and cross references, become the child entities of the master record. Note that groovy scripts on relationship objects don't run during merges. If they're critical, you can re-implement the scripts on the parent account or contact.

How Records are Selected for Automerge

Records are selected for automerge based on the following criteria:

- Score threshold: The score threshold is defined in the Match Configuration and determines if a record is included in a duplicate set.
- Automerge threshold: The automerge threshold is defined by the ZCH_AUTO_MERGE_THRESHOLD profile option and determines if the merge request for a duplicate set is processed automatically or if it must be reviewed manually.

Three possible outcomes for each record with regard to duplicate identification and merging are as follows:

- Low score below score threshold: The record isn't included in duplicate set and in the merge request for that duplicate set.
- Medium score above score threshold and below automerge threshold: The record is included in duplicate set but merge request for that duplicate set must be reviewed manually.
- High score above score threshold and above automerge threshold: The record is included in duplicate set and merge request is processed automatically.

The score for all the records in a duplicate set must be above the automerge threshold for automated processing. If one record in the duplicate set is below automerge threshold, and the other records are above the automerge threshold, the merge request must be reviewed manually.

> **Note:** When you merge two or more records with exactly same children information under phone, email, or address the children information is merged and rolled up to the master record.

How You Configure Automerge

Enabling Automerge involves several implementation steps that must be completed by an implementor using the following tasks from the Customer Data Management offering in the Setup and Maintenance work area:

- **Manage Customer Hub Profile Options**: Use this task from the Customer Hub functional area to perform the following implementation steps:
    - Set Auto Merge Threshold profile option (ZCH_AUTO_MERGE_THRESHOLD) to the required value. This profile option specifies the threshold for auto merge. Merge requests with lower scores need data steward review. An exact match is 100.
    - Review the Record Size Limit of Duplicate Set (ZCH_DI_MERGEREQ_REC_SIZE). This profile option determines the maximum number of records in the duplicate set that can be merged automatically. By default, the maximum number is set to 10 records.
    - Set the Survivorship Enabled profile option (ZCH_ENABLE_SURVIVORSHIP) to Yes. This profile option enables the survivorship rules to select the master record and retain the attributes during a merge operation.

- **Manage Survivorship Rules**: Use this task from the Customer Hub functional area to create Set Master survivorship rules to choose the master record for merge requests created from the duplicate identification batch and set the rule to active.

  If there are no active Set Master rules or if the Set Master rules didn't trigger, the merge request must be reviewed manually, even if the ZCH_AUTO_MERGE_THRESHOLD profile option is set, the score for all records is above the threshold value, and the number of records is below the record size limit.

  > **Note:** You can use the Set Attribute rules with Set Master rules to determine the Golden Master record. For automerge, Set Master rule is mandatory.

- **Manage Enterprise Data Quality Matching Configurations**: Use this task from the Data Quality Foundation functional area to perform the following implementation steps:

  - Create an active Match Configuration in Manage Enterprise Data Quality Matching Configurations task or use a predefined Match Configuration. Rebuild the keys if necessary.
  - Enable EDQ Real Time and Batch Basic Match Server in Manage Server Configurations task.

Run Automerge

This task involves the following two steps:

1. Create a duplicate identification batch and select Create Merge Request as the Automatic Processing Option.
2. Perform the task Run Request Dispatch Job to disposition the duplicate resolution sets.

The Dispatch Job processes any resolution request in Pending or Submitted status. You can run this job in two modes:

- On demand: **Run Request Dispatch Job** > **Submit**
- Per a specific schedule: Do the following steps to set up a recurring job:

  a. Click **Advanced** on the **Run Request Dispatch Job**task.
  b. Click **Schedule** tab and select the **Using a Schedule** radio button.
  c. Select the frequency you want and click **Submit**.

To see the list of dispatch jobs, and their statuses, navigate to **Scheduled Processes** under **Tools**.

Troubleshoot Automerge Issues

After you create your Duplicate Identification Batch, drill down into the completed batch to see the results. If duplicate sets have been found, and automerge is enabled, resolution requests are automatically submitted for merge.

If the resolution request wasn't submitted automatically, you can drill down to the duplicate set and compare the score for each record with the threshold in the ZCH_AUTO_MERGE_THRESHOLD profile option and the number of records with the limit in the ZCH_DI_MERGEREQ_REC_SIZE profile option. If all scores are above the threshold and the number of records is below the limit, verify that the following are true:

- Set Master rules are active and triggered to choose a master for the records in the duplicate set.
- ZCH_ENABLE_SURVIVORSHIP is set to yes.

High Volume Batch Deduplication

Batch deduplication of account or contact records in Oracle Customer Data Management Cloud Service by duplicate identification or resolution.

Batch deduplication consists of the following two steps:

- Duplicate Identification: This step includes the identification of duplicate records by submitting a Duplicate Identification Batch job.

  You can define and submit this job from the Duplicate Identification page.
- Duplicate Resolution: This step includes the resolution of the duplicates, typically by merging each set of duplicate records.

  You can resolve the duplicates either automatically by submitting the Duplicate Identification Batch job (called Automerge) or manually by submitting records in bulk from the Duplicate Identification Batch results review page.

For more details on these steps and for configuration of Automerge, see Merge Requests, Implementing Customer Data Management.

Both of these jobs are data-intensive operations that can read or update millions of rows of data in various Oracle Application Cloud tables. This document is intended to provide the guidelines and best practices for planning the data-sets, and applying appropriate configurations to achieve optimal throughput for high volume deduplication in Oracle Customer Data Management Cloud Service. Each customer's data set is unique. The time required to process a duplicate identification batch varies on the data shape.

Best Practices for High Volume Batch Deduplication

Customer Data Management merge is a data-intensive process that scans and updates a large number of tables in Oracle Applications Cloud, to correctly merge two or more Accounts or Contacts.

This section describes how you can use the following profile options to optimize the merge process:

- **Scope of Merge Process** (ORA_ZCH_MERGE_SCOPE): You can use this profile option to define the scope of the merge process.
- **Master Record Selection Method** (ORA_ZCH_SETMASTER): You can use this profile option to specify the method for selecting the master record in a merge request.
- **Create Automerge with Review** (ORA_ZCH_AUTOMERGE_REVIEW): You can use the profile option to select an appropriate processing option for Automerge.
- **Maximum Number of Concurrent Merge Jobs** (ORA_ZCH_MERGE_MAX_REQUEST_LIMIT): Specify the maximum number of merge jobs to be processed at a time. If you don't set the maximum limit, all merge jobs are submitted for concurrent processing.

You can set these profile options in the Setup and Maintenance work area using the following:

- Offering: Customer Data Management
- Functional Area: Customer Hub
- Task: Manage Customer Hub Profile Options

How You Define the Scope of the Merge Process

When you merge two or more records, the application scans hundreds of transactional and reference tables across all modules in the Oracle Applications Cloud such as, Core Customer Data Management, CRM, Financials, and Manufacturing. This can make merge a data-intensive and time consuming process. However, you can use the Scope of Merge Process (ORA_ZCH_MERGE_SCOPE) profile option to define and limit the scope of merge process in an implementation so that the application scans only the necessary business areas. This optimizes the size of the merge memory and execution profile.

The following options are supported by the Scope of Merge Process profile option:

- All Functional Areas (ALL): This is the default option and scans across all areas of Oracle Applications Cloud. You use this option when there's a global implementation running various modules of Oracle Applications Cloud such as, Core Customer Data Management, CRM, Financials, and Manufacturing.
- All Customer Relationship Management Related Areas (CRM): This option limits the scope of the process to handle all the CRM entities such as, Opportunities, and Leads, core Customer Data, Common Entities such as, Notes, and Activities, and Custom Objects. You use this option when there's a CRM implementation along with the use of Customer Data Management functionality.
- Customer Data Management Specific Areas: This option limits the scope of the process to core Customer Data, Common Entities such as Notes and Activities, and Custom Objects. You use this option during the initial customer data consolidation and to achieve best performance for Customer Data management, implementations.

Note: The profile option settings can be changed at any time, if additional modules are turned on the instance. For instance, the Customer Data Management option might be used during initial consolidation and cleanup of customer data and then changed to CRM or ALL options if other modules are implemented later.

How You Define the Master Record Selection Method

The performance of the merge process also depends on the method used to select the master record. You can use the Master Record Selection Method (ORA_ZCH_SETMASTER) profile option to specify an appropriate option for selecting the master party automatically during merge. The following options are supported by the Master Record Selection Method profile option:

- Select master record using survivorship rule (RULE): This is set as the default master selection option. This option selects the master record based on the Set Master rules defined in the Manage Survivorship task. These rules are applied using the Oracle Business Rules component. You use this option when there are complex business rules required to pick the master.
- Select the oldest record as master (OLDEST): This option selects the party with the earliest creation date as the master.
- Select the newest record as master (NEWEST): This option selects the party with the newest creation date as the master.
- Select master based on duplicate identification results (ANY) - This option randomly selects one of the parties in the set as a master.

How you Configure Automerge Action

Automerge is the process of automatically merging identified duplicate sets that exceed the automerge threshold. The process is initiated by creating a duplicate identification batch with the Create Merge Request option. You can use the

Create Automerge with Review (ORA_ZCH_AUTOMERGE_REVIEW) profile option that has Yes and No values to select an appropriate processing option for Automerge:

- Create merge requests only for duplicate sets exceeding the automerge threshold: To enable this processing option, select No as the value for the Create Automerge with Review (ORA_ZCH_AUTOMERGE_REVIEW) profile option. If you select this option, the application processes duplicate sets as follows:
    - The application preprocesses the duplicate sets exceeding the automerge threshold and merges them into a single job. This option is ideal for processing high volumes of merge requests when the duplicate sets require no review or any further action.
    - Duplicate sets not exceeding the automerge threshold remain in Not Reviewed status in the Duplicate Identification page, from where they can be manually converted to merge requests, or rejected, if needed.
- Create Merge Requests for all duplicate sets: To enable this processing option, select Yes as the value for the Create Automerge with Review (ORA_ZCH_AUTOMERGE_REVIEW) profile option. If you select this option, merge requests are created for all duplicate sets. All requests are first pre-processed. Then they're either merged (if they exceed the automerge threshold), or put in "New" status (so that they can be reviewed) if they don't exceed automerge threshold.

How you Control the Concurrency of Merge Processes

Each merge request executes as a single batch process in the Enterprise Service Scheduler (ESS). The number of merge requests executing concurrently is limited by the number of batches being concurrently processed. Therefore, if there are other ESS processes competing for threads when there are a large number of merge requests queued up, then the scheduling of those jobs could get delayed.

During initial consolidation of customer data, it's advantageous to use the maximum available threads. However, in steady state when there are other processes running in the background, it may be necessary to limit and control the number of concurrent merge ESS jobs.

To achieve this, set the following profile option to an appropriate value:

- **Profile Option Name**: Maximum Number of Concurrent Merge Jobs
- **Profile Option Code**: ORA_ZCH_MERGE_MAX_REQUEST_LIMIT
    - When the profile option value is left blank or when no value is defined, the ESS will allocate merge requests according to the threads available. This is recommended during initial high volume data processing.
    - After initial data load, set the profile option value to ten or lower if other processes such as Web services or other ESS jobs are running.

FAQs for Duplicate Resolution

Why didn't I see custom objects while creating a merge request?

The merge process handles custom objects that have been published to the mainline repository; it doesn't handle custom objects in unpublished sandboxes.

Is there a limit to the number of characters in an API name of custom fields in a duplicate resolution request?

Yes, there is a character limit. Ensure that the API name of custom fields in a duplicate resolution request is 100 characters or less.

How can we exclude custom Groovy validations and triggers during execution of merge?

You can add the following Groovy code to exclude validations during execution:

```
def username = adf.context.getSecurityContext()?.getUserName();
if(userName!='FUSION_APPS_CRM_ESS_APPID' &amp;&amp;
userName!='fusion_apps_crm_ess_appid' &amp;&amp;
userName!='FUSION_APPS_CRM_SOA_APPID' &amp;&amp;
userName!='fusion_apps_crm_soa_appid')
{

}
```