Oracle FCCM Cloud Service

Matching Guide

Release 23.11.1

November 2023

F48798-01

Getting Started with OFS Cloud Service Matching Guide

Document Control

Table 1: Document Control

Revision Number	Revision Date	Change Log
23.11.1	November 2023	There were no updates to this document in this release.
23.8.1	August 2023	Added Transliteration chapter.
23.5.1	May 2023	There were no updates to this document in this release.
23.2.1	March 2023	There were no updates to this document in this release.
22.11.1	November 2022	There were no updates to this document in this release.
22.6.1	June 2021	This is created for the version v21.6.1.0.0 release.

Contents

1 Introduction

Oracle Financial Crime and Compliance Management Cloud Service makes use of the Cloud Matching Service, which provides a flexible and customizable strategy for matching customer records to watch list records. Sanctions screening typically requires the business to employ tightly-defined, zero-tolerance matching policies that will identify every possible match against a sanctions list. In these cases, the additional review work of lower probability matches will be necessary. By contrast, a business carrying out PEP screening may choose a strategy of finding and investigating only the most likely matches against the PEP list, and the additional work required to confirm or eliminate weaker matches may not be cost-effective for the business. Oracle Financial Crime and Compliance Management employs matching rules widget to configure the rules for screening. These can be enabled and disabled as needed, to tune the behavior of Oracle Financial Crime and Compliance Management Cloud Service to your requirements. The matching rules are built around name matching. Other identifiers are also used in the matching rules, but their main purpose is to rank matches by strength, and thereby to enable a most-likely approach to review potential matches.

For example, strong matches to Sanctions lists should be regarded as the most urgent matches, requiring immediate attention. Strong matches to PEP records will require follow-up, but may not be so urgent. Looser matches to PEP records may not be worth the time and operational cost of review. In general, the looser the match rule, the more likely it is to raise false positives. It is not possible to eliminate all false positives, especially if there is a requirement to identify all true matches. Tuning the matching strategy is, therefore, a trade-off between the proportion of true matches that are not detected and the work required to manually eliminate false positives. This will be evident in the examples in this document.

2 General matching strategy

This section provides a brief description of the general strategy used in Oracle Financial Crime and Compliance Management Cloud Service. It consists of three main components: identifier preparation, clustering, and matching.

2.1 Identifier preparation

There are some differences between the structure of data sets that always need to be normalized before clustering and matching, so that the matching process does not need to repeat the configuration of transformations on each comparison.

Identifier preparation is used to ensure that the records conform to a pre-defined data structure which can be used by the rest of the matching process, and to eliminate common forms of variance between the records (such as spelling variants of given names and abbreviations of frequently used tokens).

2.2 Clustering

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

If very wide clusters are used, there will be a large number of records in each cluster. This means that there is a reduced risk that true matches will be missed, but also that a greater amount of processing power is required to compare all the clustered records by brute force. A tighter clustering strategy will result in smaller clusters, with fewer records per cluster. This results in reduced processing requirements for row-by-row comparisons but increases the likelihood that some true matches will not be detected.

2.3 Matching

Once the working and watch list records have been divided into clusters, the rows within each cluster are compared to one another according to the match rules defined for the matching processor. Each match rule defines a set of criteria, specified as comparisons, that the pair of records must satisfy in order to qualify as a match under that rule. The match rule also defines a decision to be applied to any records which satisfy the conditions of the rule. Most rules have a Review decision, meaning matches that hit the rule need to be reviewed. However, there are also elimination rules, where if the records being compared meet the rule's criteria, a No Match decision is reached, and the two records will not be considered a match.

The rules are applied as a decision table, so if a pair of records qualifies as a match under a rule higher in the table, it will not be compared using any rules below that. All rules are configured to operate on a case-insensitive basis. Unless stated otherwise, all noise and whitespace characters are removed or normalized before matching.

Matches are generated based on a defined set of attributes for each rule. A weighted average of the score is generated for each of the attribute level matches.

There are two types of matching services:

- Real-Time query processing
- Bulk query processing

In Real-Time query processing, a string value given in the UI is matched against a column in the target table. The screening application explicitly passes the strings as values in the request which forms “the strings to be matched” against “all the values in a column name”. Then, based on the matches received for the source string from search engine, the score and the feature vector for the matched strings (source and target) are generated. Scores which exceed the configured thresholds are taken and collected.

Example 1:

Consider the following match details:

- Evaluation Logic-> 46,4,26
- Evaluation score -> 0.88
- Word Match count (WMC_1) >= 2
- Abbreviated and CMP >=66

If the first name in **Mapping Source Attribute** column and **Target Attribute** column matched with a score of 0.88 and the Set threshold value is 0.75. The first name score is greater than the threshold value. Then 0.88x100x weightage (0.8) is provided in matching rule which gives you the score -> 70.4.

If **Mapping Source Attribute** column and **Target Attribute** column has the city data and the score value doesn't cross the provided threshold then it will not contribute to the score. Score is still 70.4

If city data cross the provided threshold, then it will contribute to the score as 100 (exact match). Then 100x0.05x weightage (0.05) => 5 is provided in matching rule which gives you the score 75.8.

If **Mapping Source Attribute** column and **Target Attribute** column has no city data, then score will cross the provided threshold. Then 50x0.05 => 2.5 is provided in matching rule which gives you the score 72.9.

Similarly, if other column data matches and score crosses the threshold, then the score will be added

Table 1 provides Match Types Descriptions and Examples:

Table 1: Match Types Descriptions and Examples

Ruleset Name	Source Node Type	Target Node Type
Exact	Considers two values and determines whether they match exactly. Applies only if Exact Match is selected. It does not apply when using Fuzzy Match.	If the source attribute is “John smith” and target attribute is “John smith”, then the match is an exact match.

Table 1: Match Types Descriptions and Examples

Character Edit Distance (CED)	Considers two String tokens and determines how closely they match each other by calculating the minimum number of character edits (deletions, insertions and substitutions) needed to transform one value into the other.	For entities, stop words are not considered. ● If the source attribute is “John smith” and target attribute is “Jon smith”, then the CED is 1 since the letter 'h' is missing between the source attribute and target attribute. ● If the entity names are Oracle Financial Corporation and Finance Orcl Pvt. Ltd., then only Oracle Financial and Finance Orcl are considered for matching as corporation, Pvt., and Ltd. are stop words. The CED for Orcl is 2 and CED for finance is 3, so the overall CED is 3.
Character Match Percentage (CMP)	Determines how closely two values match each other by calculating the Character Edit Distance between two String tokens and considering the length of the shorter of the two tokens, by character count.	If the source attribute is “John smith” and target attribute is “Jon smith”, then the CMP is calculated using the formula (length of shorter string – CED) * 100 / length of longer string. In this case, it is (9-1) * 100/8 = 77.77%.

Table 1: Match Types Descriptions and Examples

Word Edit Distance (WED)	Determines how well multi-word String values match each other by calculating the minimum number of word edits (word insertions, deletions and substitutions) required to transform one value to another.	If the source attribute is "John smith" and target attribute is "Jon smith", then the WED is calculated by checking the number of words that did not match with the target words after allowing for character tolerance, which is the number of words in the source attribute that did not match the target attribute. For example, the source string is Yohan Russel Smith and target string is Smith Johaan Rusel. First, we determine the CED for each word: ● Yohan matches with Johann with a CED of 2. ● Russel matches with Rusel with a CED of 1. ● Smith matches with Smith with a CED of 0. If we consider a character tolerance of 1, we can observe the following: ● Russel with a character tolerance of 1 match with Rusel. ● Smith with a character tolerance of 0 matches with Smith. ● Yohan with a character tolerance of 2 does not match with Johann as the character tolerance is 1. Based on these observations, we can conclude that one word does not match. This means that the WED is 1.
Word Match Percentage (WMP)	Determines how closely, by percentage, two multi-word values match each other by calculating the Word Edit Distance between two Strings and taking into account the length of the longer or the shorter of the two values, by word count.	The WMP is calculated using the formula (WMC/minimum word length) * 100. If the source attribute is "John smith" and target attribute is "Jon smith", then the WMP is calculated as (2/5) * 100 = 40 %.
Word Match Count (WMC	Determines how closely two multi-word values match each other by calculating the Word Edit Distance between two Strings and taking into account the length of the longer or the shorter of the two values, by word count.	The WMC is like WED, with the difference being that WMC gives the number of matches between 2 words and WED gives the number of words that did not match between 2 words. If the source attribute is "John smith" and target attribute is "Jon smith", then the WMC is 2 as two words have matched (allowing for the character tolerance).
Exact String Match	Considers two String values and determines whether they match exactly.	

Table 1: Match Types Descriptions and Examples

Abbreviation	Checks if the first character matches with the first character of source and target values.	
Starts With	Compares two values and determines whether either value starts with the whole of the other value. It therefore matches both exact matches and matches where one of the values starts the same as the other but contains extra information.	
Jaro Winkler or Reverse Jaro Winkler	The Jaro Winkler similarity is the measure of the edit distance between two strings. Click here for more information. In the Reverse Jaro Winkler, matches are generated even if the string is reversed. For example, if the source string is Mohammed Ali and the target string is Ali Mohammed, then the similarity = 1.	If the source string is Mohammed Ali and the target string is Mohammed Ali, then the similarity = 1.
Levenshtein	The Levenshtein Distance (LD) or edit distance provides the distance, or the number of edits (deletions, insertions, or substitutions) needed to transform the source string into the target string. Click here for more information.	For example, if the source string is Mohamed and the target string is Mohammed, then the LD = 1, because there is one edit (insertion) required to match the source and target strings.

Example 2:

Source String: Mrs. Eisa Carrera Ben

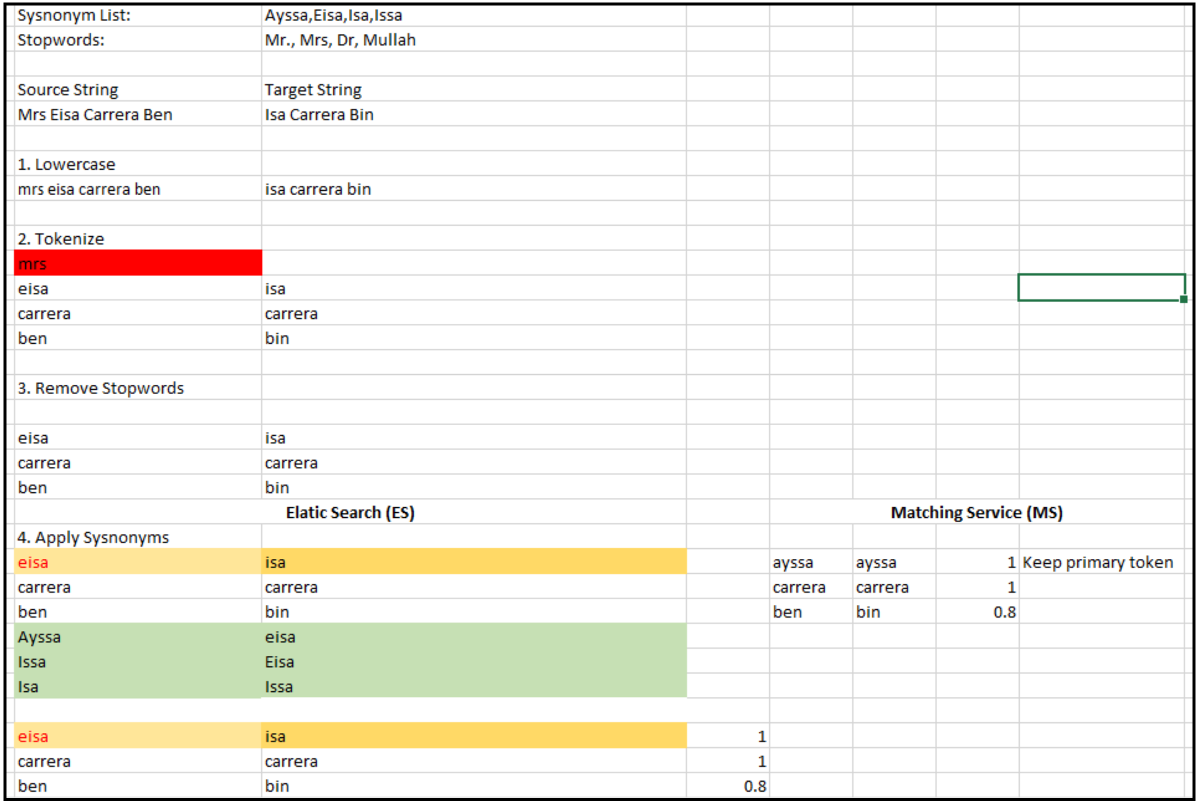
Target String: Isa Carrera Bin

Synonym List: Ayssa, Eisa, Isa, Issa

Stopwords: Mr., Mrs., Dr., Mullah

Table 1 illustrates the synonyms and stopwords for OS and MS.

Figure 1: Synonyms and Stopwords for OS and MS



Sysnonym List:	Ayssa,Eisa,Isa,Issa					
Stopwords:	Mr., Mrs, Dr, Mullah					
Source String	Target String					
Mrs Eisa Carrera Ben	Isa Carrera Bin					
1. Lowercase						
mrs eisa carrera ben	isa carrera bin					
2. Tokenize						
mrs						
eisa	isa					
carrera	carrera					
ben	bin					
3. Remove Stopwords						
eisa	isa					
carrera	carrera					
ben	bin					
	Elatic Search (ES)		Matching Service (MS)			
4. Apply Sysnonyms						
eisa	isa		ayssa	ayssa	1	Keep primary token
carrera	carrera		carrera	carrera	1	
ben	bin		ben	bin	0.8	
Ayssa	eisa					
Issa	Eisa					
Isa	Issa					
eisa	isa	1				
carrera	carrera	1				
ben	bin	0.8				

A sample score is mentioned in the example for illustration.

3 Scoring Method

The scoring methods used in the entity resolution component are as follows:

- Jaro Winkler
- Levenshtein
- Individual Name
- Individual SAN
- Individual PEP
- Individual EDD
- Entity Name
- Entity SAN
- Entity PEP
- Entity EDD

3.1 Jaro Winkler

This algorithm gives high scores for the following strings:

The strings contain the same characters but within a certain distance from one another.

The order of the matching characters is the same.

To be precise, the distance of finding a similar Character is one Character less than half of the length of the longest string. So if the longest string has a length of five, a character at the start of string 1 must be found before or on ((5/2)–1) ~ 2nd position in the string 2. This is considered a good match. Hence, the algorithm is directional and gives a high score if matching is from the beginning of the strings.

For example:

- `textdistance.jaro_winkler("mes", "messi") 0.86`
- `textdistance.jaro_winkler("crate", "crat") 0.96`
- `textdistance.jaro_winkler("crate", "atcr") 0.0`

In the first case, as the strings match from the beginning, a high score is given. Similarly, in the second case, only one Character was missing, and that is also at the end of string 2, a very high score is given. In the third case, the last two Characters of string 2 are rearranged by bringing them at the front, resulting in 0% similarity.

3.2 Levenshtein

The Levenshtein Distance (LD) or edit distance provides the distance or the number of edits (deletions, insertions, or substitutions) needed to transform the source string into the target string. For example, if the source string is Mohamed and the target string is Mohammed, the LD = 1 because one edit (insertion) must match the source and target strings. For more information, see the website.

3.3 Individual Name

This is a bespoke rules-based algorithm that has been optimized for determining individual name matches.

Generation of the final matching score for **Individual Name** is based on the combination of scores generated by the following feature vector in the scoring method:

- **Abbreviation**: Checks if the first Character matches with the first Character of source and target tokens.

  Example:

  - **String 1**: "S Turner"
  - **String 2**: "Steve Turner"

  String1 is an abbreviated string for string2.

- **Character Edit Distance (CED):** The Character Edit Distance comparison compares two String tokens and determines how closely they match each other by calculating the **minimum,** and the **maximum** number of character edits (deletions, insertions, and substitutions) needed to transform one value into the other.

  It compares each token in a string with each token of another string and finds the minimum edits we need to convert one token to another.

  Maximum CED score that is required to convert one token to another token in a string.

  The stopwords for Individual Names are Mr., Mrs, and Ms.

  Example:

  - **String1**: “Mr Jerrod Benito Carrera”
  - **String2**: “JOSE BENITO CABRERA”
  - **Stopword**: Mr.

  Jerrod matches with JOSE → CED: 5 (CED_MAX))

  Benito matches with BENITO → CED: 0 (CED_MIN)

  Carrera matches with CABRERA → CED: 1

- **Character Match Percentage (CMP):** The Character Match Percentage comparison determines how closely two values match each other by calculating the Character Edit Distance between two String tokens and taking into account the shorter length of the two tokens by character count.

  CMP = (MCL – CED) * 100 / MCL

  - **CMP** = Character match percentage
  - **MCL** = Maximum character Length

- **Exact String Match:** The Exact String Match comparison is a simple comparison that determines whether or not two String values match.

  Checks if all the tokens are exact to each other. It can be in any order.

  Example:

  - **String 1:** Ram Lakshaman
  - **String 2:** Lakshaman Ram
  - **String 3:** Ram Lakshaman

  String 1, String 2, and string 3 are exact.

- **Starts With:** The Starts With comparison compares two values and determines whether either value starts with the whole of the other value. It, therefore, matches both exact matches and matches where one of the values starts the same as the other but contains extra information.

Checks if all the tokens start with the respective token in the target tokens.

> **NOTE** Whichever token is smaller (either in source or target), that token will be considered compared with the other token (of longer length). It should be in order.

- **Starts With First:** It is similar to **Starts With**. Starts with the first token only.
- **Metaphone:** Checks if strings sound like. It is similar for hearing, but spelling may be different. It encodes a string into a double Metaphone value.
- **Tokenize Jaro:** Checks similarity is the measure between two strings. It tokenizes source and target strings, then uses the **Jaro Winkler** algorithm to calculate the score between tokens, and then consolidates the scores to a single score by taking the average.

  Word Edit Distance (WED): The Word Edit Distance comparison determines how well multi-word String values match each other by calculating the minimum number of word edits (word insertions, deletions, and substitutions) required to transform one value to another.

  So WED is similar to CED, where instead of character edits, we find the word edits.

  In WED, we have an additional parameter called "character tolerance." Character tolerance allows the user to have a character tolerance in words, i.e. how many Character edits can it allow in each token for one token to match another one.

  WED in simple words is: Number of words that did not match with the target words (after allowing the character tolerance)

  Example:

  - **String 1**: "Yohan Russel Smith"
  - **String 2**: "Smith Johaan Rusel"

  Yohan matches with Johann → CED: 2

  Russel matches with Rusel → CED: 1

  Smith matches with Smith → CED: 0

  - If we have a character tolerance of "1,". The number of WED will be: 1
    - Russel, with a character tolerance of 1 matches with Rusel.
    - Smith with character tolerance of 0 matches with Smith.
    - Yohan with character tolerance of 2 does not match with Johann as character tolerance is 1.

    One token did not match. WED = 1.
  - If we have a character tolerance of "2,". The number of WED will be : 0
    - Russel, with a character tolerance of 1 matches with Rusel.
    - Smith with character tolerance of 0 matches with Smith.
    - Yohan, with a character tolerance of 2, does match with Johann.

  All tokens matched. WED = 0.
- **Word Match Percentage (WMP):** The Word Match Percentage comparison determines how closely two multi-word values match each other by calculating the Word Edit Distance between two Strings and considering the length of the longer or longer or longer, the shorter of the two values, by word count.

In mathematical terms, the Word Match Percentage comparison uses the following formula to calculate its results as the **Number of tokens matched**:

WMP = (WMC / WL) *100

- **WMP:** Word Match Percentage
- **MWL**: Maximum Word Length (i.e., the maximum number of words in the two values being compared)
- **WED**: Word Edit Distance between two String values, and
- **WL**: Minimum Word Length, relating to shorter input option

- **Word Match Count (WMC):** The Word Match Percentage comparison determines how closely two multi-word values match each other by calculating the Word Edit Distance between two Strings and taking into account the length of the longer or, the longer shorter of the two values, by word count.

  WMC, in simple words, is the Number of words that did match with the target words (after allowing the character tolerance).

  WMC compliments WED, as WMC gives the number of matches between 2 words and WED gives the number of mismatches between 2 words.

> **NOTE** Based on the Year of birth, if the match falls beyond +/- 5 years, then the match is eliminated. This is applicable for the Individual PEP and the EDD records.

Example:

- **String 1**: "Yohan Russel Smith"
- **String 2**: "Smith Johaan Rusel"

  Yohan matches with Johann → CED: 2

  Russel matches with Rusel → CED: 1

  Smith matches with Smith → CED: 0

- If we have a character tolerance of "1,". The number of WMC will be: 2
  - — Russel, with a character tolerance of 1 matches with Rusel.
  - — Smith with character tolerance of 0 matches with Smith.
  - — Yohan with character tolerance of 2 does not match with Johann as character tolerance is 1.

  One word did not match. So WMC = 2.

- If we have a character tolerance of "2,". The number of WMC will be : 3
  - — Russel, with a character tolerance of 1 matches with Rusel.
  - — Smith with character tolerance of 0 matches with Smith.
  - — Yohan, with a character tolerance of 2, does match with Johann.

All tokens matched. WMC = 3.

Example:

- **String 1**: “Mr Jerrod Benito Carrera”

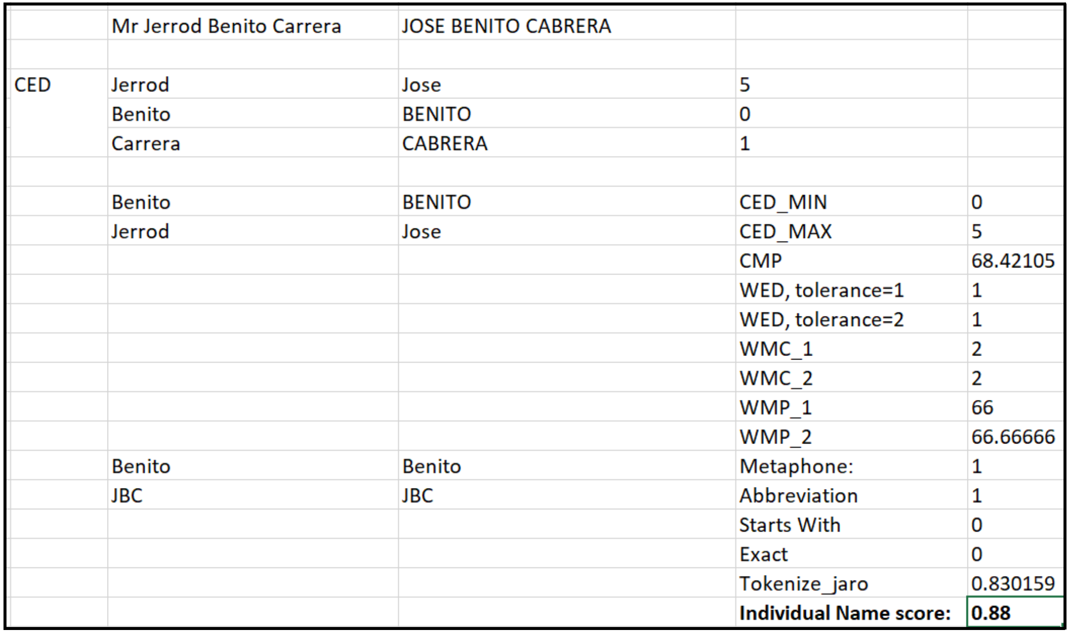
- **String 2**: “JOSE BENITO CABRERA”
- **Stopword**: Mr. (It will not be considered for calculating the score)

The individual feature vector scores with final scrore for the match:

```
{"ced_list": [5, 0, 1], "ced_min": 0, "ced_max": 5, "cmp": 68.42105, "wed_1":
1, "wed_2": 1, "wmc_1": 2, "wmc_2": 2, "wmp_1": 66.0, "wmp_2": 66.666664,
"metaphone": 1, "starts_with": 0, "abbreviation": 1, "tokenize_jaro":
0.8301587, "exact": 0, "inorderMaxPos": 0, "score": 0.88}
```

The following figure illustrates the Individual Name score generated based on the combination of scores:

Figure 2: Individual Name Score



	Mr Jerrod Benito Carrera	JOSE BENITO CABRERA		
CED	Jerrod	Jose	5	
	Benito	BENITO	0	
	Carrera	CABRERA	1	
	Benito	BENITO	CED_MIN	0
	Jerrod	Jose	CED_MAX	5
			CMP	68.42105
			WED, tolerance=1	1
			WED, tolerance=2	1
			WMC_1	2
			WMC_2	2
			WMP_1	66
			WMP_2	66.66666
	Benito	Benito	Metaphone:	1
	JBC	JBC	Abbreviation	1
			Starts With	0
			Exact	0
			Tokenize_jaro	0.830159
			Individual Name score:	**0.88**

3.4 Individual SAN

This is a bespoke rules-based algorithm that has been optimized for determining Individual Sanctions name matches.

The process and methods to generate feature vector are similar to **Individual Name**. For more details, see Individual Name.

3.5 Individual PEP

This is a bespoke rules-based algorithm that has been optimized for determining Individual PEP name matches.

The process and methods to generate feature vector are similar to **Individual Name**. For more details, see Individual Name.

3.6 Individual EDD

This is a bespoke rules-based algorithm that has been optimized for determining Individual EDD name matches.

The process and methods to generate feature vector are similar to **Individual Name**. For more details, see Individual Name.

3.7 Entity Name

This is a bespoke rules-based algorithm that has been optimized for determining organization name matches. For example, BLUE SKYE COLLECTIONS LTD

The process and methods to generate feature vector are similar to **Individual Name**. For more details, see Individual Name.

The process to derive the final score from the feature vector is different from **Individual Name.**

The stopwords are not considered for computing the score. The stopwords for Entity Name are different from **Individual Name**.

Example:

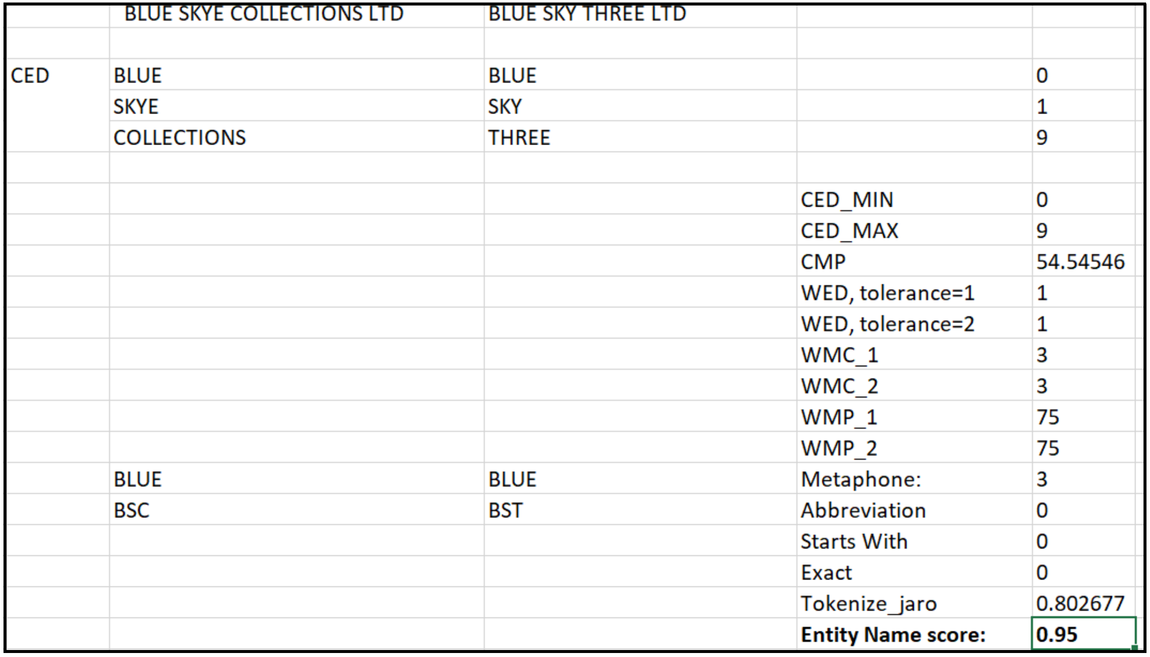
- **String 1**: "BLUE SKYE COLLECTIONS LTD"
- **String 2**: "BLUE SKY THREE LTD"
- **Stopword**: LTD (It will not be considered for calculating the scrore)

The individual feature vector scores with overall scrore for the match:

```
{"ced_list": [0, 1, 9, 0], "ced_min": 0, "ced_max": 9, "cmp": 54.545456,
"wed_1": 1, "wed_2": 1, "wmc_1": 3, "wmc_2": 3, "wmp_1": 75.0, "wmp_2": 75.0,
"metaphone": 3, "starts_with": 0, "abbreviation": 0, "tokenize_jaro":
0.8026768, "exact": 0, "inorderMaxPos": 0, "score": 0.95}
```

The following figure illustrates the Individual Name score generated based on the combination of scores:

Figure 3: Individual Name Score



	BLUE SKYE COLLECTIONS LTD	BLUE SKY THREE LTD		
CED	BLUE	BLUE		0
	SKYE	SKY		1
	COLLECTIONS	THREE		9
			CED_MIN	0
			CED_MAX	9
			CMP	54.54546
			WED, tolerance=1	1
			WED, tolerance=2	1
			WMC_1	3
			WMC_2	3
			WMP_1	75
			WMP_2	75
	BLUE	BLUE	Metaphone:	3
	BSC	BST	Abbreviation	0
			Starts With	0
			Exact	0
			Tokenize_jaro	0.802677
			Entity Name score:	**0.95**

3.8 Entity SAN

This is a bespoke rules-based algorithm that has been optimized for determining Sanctions organization name matches.

The process and methods to generate feature vector are similar to **Individual Name**. For more details, see Individual Name.

3.9 Entity PEP

This is a bespoke rules-based algorithm that has been optimized for determining PEP organization name matches.

The process and methods to generate feature vector are similar to **Individual Name**. For more details, see Individual Name.

3.10 Entity EDD

This is a bespoke rules-based algorithm that has been optimized for determining Entity EDD organization name matches.

The process and methods to generate feature vector are similar to **Individual Name**. For more details, see Individual Name.

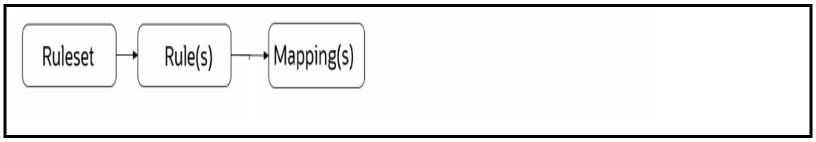
4 Rulesets

The Ruleset facilitates identifying the similarity between two entities (customer, account, and so on) and derives a match. A Ruleset is a set of rules applied to the defined source and target entities and compares the entities' attributes to derive a match.

OFS Cloud Service provides ready-to-use rulesets; however, you can modify these rulesets or create your own rulesets.

The process of creating Match Rule (s) is as follows:

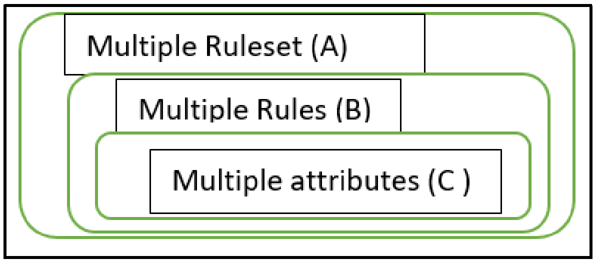
Figure 4: Match Ruleset Workflow



> **NOTE** The sum of weightage for mappings and groups for each rule must be 1.

Each Ruleset comprises multiple rules. The Ruleset compares the attributes defined in the rules for the source entity with the target entity and applies one of the match scoring mechanisms described in this document. The threshold value is defined in each level.

Figure 5: Threshold Value Levels



- Ruleset (A): All the matches from the rules should cross this threshold.
- Rule (B): All the attribute thresholds should cross this threshold.
- Attribute (C): Based on the match, it checks across the threshold. Once it crosses the threshold, the record will be considered.

4.1 Out of the Box Rulesets

This topic provides the details of Out of the Box (OOB) match and merge rulesets, Data Survival.

Topics:

- Match Rules
- Merge Rules
- Data Survival

4.1.1 Match Rules

Table 2 provides the OOB Match Rules for FSDF 808 and Global Graph Nodes.

Table 2: List of Rulesets

Ruleset Name	Source Node Type	Target Node Type
Customer To Ext Panama	Customer	Offshore
Customer To Ext Bahamas	Customer	Panama
Customer To Ext Paradise	Customer	Bahamas
Customer2Ext PanamaAddr	Customer	Paradise
Customer2ExtOffshoreAddr	Customer	ICIJ- Panama Address
Customer To Derived Entity Match	Customer	ICIJ- Offshore Address
Customer to Customer Match	Customer	Derived Entity
Derived Entity To Derived Entity Match	Customer	Customer
Customer to Customer Match	Derived Entity	Derived Entity
Customer2ExtBahamasAddr	Customer	Customer
ER - Customer to Customer Match	Customer	ICIJ- Bahamas Address
ER - Customer to Customer Match	Pre Staging - Party Data 808	Pre Staging - Party Data 808
Customer2ExtParadiseAddr	Pre Staging - Party Data 81201	Pre Staging - Party Data 81201
Customer To Ext Panama	Customer	ICIJ- Paradise Address

Each Ruleset contains the pre-defined source and target node types. Each Ruleset compares the parameters/attributes of the source and target node types to obtain a match.

4.1.2 Merge Rules

Table 3 provides OOB Merge Rules for FSDF 808.

Table 3: List of Rulesets

Ruleset Name	Source Node Type	Target Node Type
Customer to Customer Merge	Pre Staging - Party Data 808	-
ER - Customer to Customer Merge	ER - Pre Staging - Party Data 81201	-

4.1.3 Data Survival

The OOB Data Survival Rule are as follows:

- FSDF 808:
  - Rule Name: Entity Resolution – Pre-staging Party Data 808 Data Survival
  - Dataset: Customer808
- FSDF 812:

- Rule Name: Entity Resolution – Pre-staging Party Data 81201 Data Survival
- Dataset: Customer812

4.2 Create Ruleset

This section provides details on creating rulesets.

Topics:

- Match and Merge Ruleset
- Data Survival Rules

4.2.1 Match and Merge Ruleset

Each Ruleset comprises multiple rules. The Ruleset compares the attributes defined in the rules for the source entity with the target entity. For example, Customer to Customer, Customer to Derived Entity, Derived Entity to Derived Entity, and so on.

Every Entity has attributes such as email IDs, the date of birth, jurisdiction, and so on. To derive a match and create a similarity edge on the graph or merge records in FSDF, you must apply the conditions for these attributes, such as match type, scoring method, threshold score, and weightage.

4.2.2 Data Survival Rules

The Rule facilitates storing the master record/final output tables based on the dataset survival rule stored against pipeline id.

Compliance Studio provides ready-to-use rules; however, you can modify these rules or create your rules:

- Use Data Survival
- Create Data Survival

You can enable or disable the ready-to-use rules. You can also modify description and ER type attributes for these Rules and remove rulesets from the existing list.

5 Transliteration

Transliterating a word does not tell you the meaning of the word. It tells you how the how the word is pronounced in a foreign language. This makes the language a little more accessible to people who are unfamiliar with the alphabet of the foreign language. This is opposed to translation, which, put in simple terms, gives you the meaning of a word that's written in another language. For example, the greeting in Arabic is translated as شكرا in the Arabic script but is transliterated in the Latin script as shukraan.

General transforms provide a general-purpose package for processing Unicode text. They are a powerful and flexible mechanism for handling a variety of different tasks, including:

- Uppercase, lowercase, or title case conversions
- Normalization
- Hex and character name conversions
- Script to script conversion

The reference data sources supported by Customer Screening are all provided in the Latin character set, and some in the original scripts. The screening process can also be used with non-Latin data. Non-Latin data can be screened against the Latin reference data sources which are supported by performing transliteration of data from the non-Latin character set to the Latin character set.

Non-Latin customer data can be screened against non-Latin reference data without any changes to the product, although certain fuzzy text matching algorithms may not be as effective when used to match data with the non-Latin character set. Text is processed on a left-to-right basis.

Topics:

- Original Script Matching
- Input fields for Individual screening
- Input fields for Entity screening

5.1 Original Script Matching

To match the original script data against reference data, follow these steps:

1. Prepare customer and external entity data such that non-Latin names are populated
2. in the Original Script Name fields.

5.2 Input fields for Individual screening

This section lists the REST input fields used when screening individuals via the real-time process. The following input attributes are available for the individual screening process. They are available for any additional inputs required by your screening process.

Table 4 provides the list of Input fields for Individual screening.

Table 4: Input fields for Individual screening

Field Name	Expected Data Format	Notes
v_given_name	String	The individual matching process is based primarily on the name supplied for the individual.
v_family_name	String	
v_full_nm	String	
v_aliases_family_name	String	
v_aliases_given_name	String	
v_aliases	String	

5.3 Input fields for Entity screening

This section lists the REST input fields used when screening entities via the real-time process. The following input attributes are available for the entity screening process. They are available for any additional inputs required by your screening process.

Table 5 provides the list of Input fields for Entity screening.

Table 5: Input fields for Entity screening

Field Name	Expected Data Format	Notes
v_org_nm_bus_strip	String	The entity matching process is based primarily on the name supplied for the entity.
v_last_nm	String	
v_full_nm	String	
v_org_nm	String	
v_alias_nm	String	
v_aliases	String	
v_first_nm	String	

OFSAA Support Contact Details

Raise a Service Request (SR) in My Oracle Support (MOS) for queries related to OFSAA applications.

Send Us Your Comments

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, indicate the title and part number of the documentation along with the chapter/section/page number (if available) and contact the Oracle Support.

Before sending us your comments, you might like to ensure that you have the latest version of the document wherein any of your concerns have already been addressed. You can access My Oracle Support site which has all the revised/recently released documents.