Enabling global identity
Protecting digital trust

# GLEIF Check for Duplicates

## Dictionary

Public
Version 1.2 final
2021-11-10

# Table of Contents

| | |
|---|---|
| **Version** | 1.2 final |
| **Date of version** | 2021-11-10 |
| **Created by** | GLEIF |
| **Approved by** | *Head of Data Quality Management and Data Science* |
| **Confidentiality level** | Public |

# About this Document

This document contains a short description of GLEIF's Check for Duplicates facility providing more transparency to the users. This document does not replace the Implementation Note for Check for Duplicates or its Appendix.

Chapter 1 is about the purpose of the Check for Duplicates facility. Chapter 2 describes the current (as of the date of distribution of this document) approach for duplicates detection. The algorithm and parametrization may change in the future due to continuous improvement.

# Change History

This section records the history of all changes to this document.

| Date | Version | Description of change | Author |
|---|---|---|---|
| 2021-11-10 | 1.2 | Update layout and wording | GLEIF |
| 2019-09-26 | 1.1 | Clarifications | GLEIF |
| 2016-10-06 | 1.0 | Final version | GLEIF |

# Glossary

| Term | Definition |
|---|---|
| Cosine similarity | A vector similarity metric that treats each sequence as a vector, and calculates their cosine as the similarity score. |
| Exclusivity violation | A violation of this type appears when any particular Legal Entity has sufficiently similar Reference Data against the threshold. |
| Extended Exact Match | String matching technique using the normalized strings and performing exact match between those. For funds, this method is also used for partial prefix/suffix matching. |
| LEI Database | The Global LEI Repository and other LEI Records that have been submitted to the Check for Duplicates facility. |
| LE-RD | Legal entity's reference data. |
| Levenshtein distance | A metric for distinguishing differences between two strings. This distance is the number of single character edits needed to turn one string into the other. These edits consider insertions, deletions or substitutions. |
| Monge-Elkan (Cosine) distance | A string similarity metric that uses tokens (i.e. words) combined with internal distance. In this case, the internal distance metric is Cosine. |
| n-gram | Splits of any particular words in strings of length n. |
| :punct: | Posix Character Class containing a defined set of punctuation characters. |
| Uniqueness violation | A violation of this type appears when an LEI code appears more than once. |

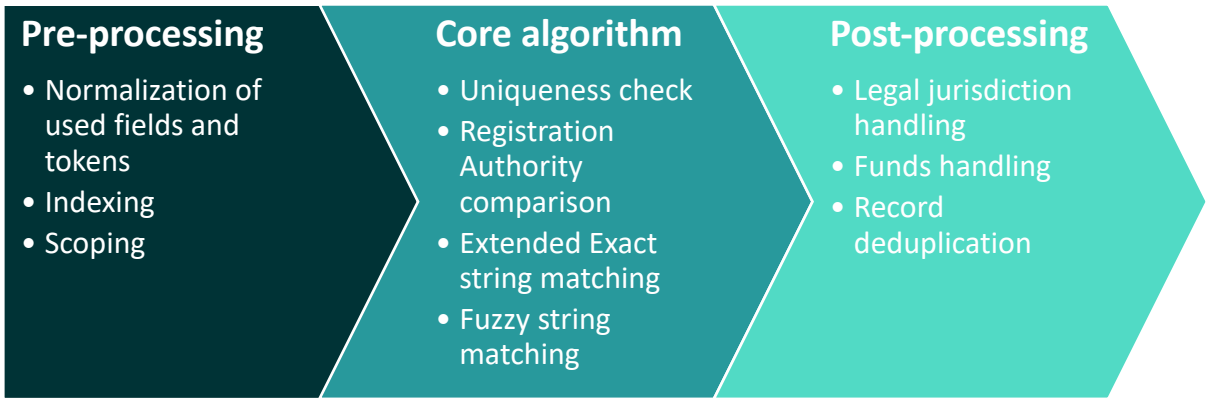
# 1. Purpose of the Check for Duplicates

In order to support the LEI issuing organizations, GLEIF provides the appropriate and mandatory processes and a technical interface for LEI Issuer to check the LEI and LE-RD data for duplicate entries. LEIs are to be checked one at a time prior to their publication using an automated web service API ("Endpoint") provided by GLEIF.

Additional to LEI Issuers' own systems checking against duplicates, LEI issuing organizations are requested to use GLEIF's Check for Duplicates facility to check all LEI Records independent of their Registration Status prior to publication. To further prevent duplicated entries, new records are compared to all records in the Global LEI Repository as well as to not yet issued LEI Records submitted to the Check for Duplicates facility by other LEI Issuers. This ensures that even if two organizations have been approached by the same legal entity, the LEI Issuers will notice the parallel process and will coordinate further with their client and the involved LEI Issuer, in order to prevent the introduction of duplicates in the system.

# 2. Check for Duplicates workflow and methodology

The general workflow of Check for Duplicates consists of three main steps:




The pre-processing contains a series of preparation activities on the data fields for the actual comparison, e.g., identified legal form terms in names are standardized and all strings are handled case-insensitive. This step also includes some technicalities like Indexing and Scoping, needed for the core algorithm and duplicates identification. Indexing and Scoping are technical details of the underlying database system and will not be further described in this document.

The core functionality of Check for Duplicates for string matching is based on three similarity algorithms: Levenshtein, Cosine and Monge-Elkan (combined with Cosine). Similarity is calculated for all combinations of LegalName, OtherEntityNames and TransliteratedNames.

In order to prune the suspected duplicates and keep the false positives at a minimum, additional measures are taken along the complete process, but especially in the Post-processing section of the workflow.

## 2.1 Pre-processing

The submitted LEI Record is always checked against the Global LEI Repository created from the latest GLEIF concatenation file. Only LEIs with EntityStatus ACTIVE and RegistrationStatus different from DUPLICATE and ANNULLED are considered for the further processing. Afterwards Indexing and Scoping is taking place.

Following preparation activities are performed:

- Ignore a leading or trailing “the”
- Handle equally “&” and “and”
- Ignore multiple consecutive spaces
- Handle equally different spellings of the same legal form (e.g.: “Société anonyme”, “SA” or “S.A.”)
- Ignore punctuation characters as defined in :punct: (Extended Exact Match and Registration Authority comparison only)

Different steps in the core algorithm consider different normalization steps from the above describe list. In general, Check for Duplicates is also setup in a way to operate case-insensitive and (tokens) order insensitive.

## 2.2 Core algorithm

**Step 1: Uniqueness check**

First, a uniqueness check is performed and records with identical LEIs to the submitted LEI Record are returned as Uniqueness violations in the response with "duplicate_type": "UNIQUENESS". Regardless of the result of this check, the next steps are also performed for all records.

**Step 2: Registration Authority comparison**

An exact match on the RegistrationAuthorityID (RAID) and normalized RegistrationAuthorityEntityID (RAEID) will be directly (i.e.: without additional comparison of the names) suggested as a potential duplicate in the response with “duplicate_type”: EXCLUSIVITY. Likewise, an exact match on the (Other)ValidationAuthorityID ((O)VAID) and normalized (Other)ValidationAuthorityEntityID ((O)VAEID) will also result in a potential duplicate. Reserved codes for RegistrationAuthorityID and ValidationAuthorityID (RA888888 and RA999999) are excluded as well as missing RAEID and VAEID codes.

Example 1: Registration Authority comparison

| LEI sent to the facility | LEIs in the Database | Potential Duplicates |
|---|---|---|
| RAID: RA123456<br>RAEID: AB-123.456.789 | LEI A: RAID: RA123456<br>RAEID: AB123456789<br><br>LEI B: RAID: RA654321<br>RAEID: AB-123.456.789 | **LEI A** |

Note: LEI Issuers should avoid publishing “n/a” (or similar) as RAEID for registries that do not provide an entity ID as this would count as exact match and potentially return a large number of duplicate records. The usage of placeholder values like “n/a” is monitored via GLEIF’s Data Quality Checks.

**Step 3: Extended Exact string matching**

In general, names used for comparison are taken from the fields LegalName, OtherEntityName and TransliteratedOtherEntityName with all possible combinations.

The first part in the name matching algorithm is using Extended Exact matching. If the names match, the records are presented as potential duplicates to the user.

Example 2: Extended Exact matching

| LEI sent to the facility | LEIs in the Database | Potential Duplicates |
|---|---|---|
| LegalName:<br>YES Bank Inc. | LEI A: LegalName:<br>yes-bank, Inc<br><br>LEI B: LegalName:<br>The Bank LTD | **LEI A** |

**Step 4: Fuzzy string matching**

If the names do not match in Step 3, three fuzzy string-matching algorithms are performed after applying the following pre-processing on each original name:

- Pre-processing steps as described above
- Remove weak tokens and store them in separate weak token lists for each string

After this, two strings (i.e.: name of the submitted record and name in the LEI Database) are compared:

1. Levenshtein – This is the simplest similarity algorithm, which checks two strings for insertions, substitutions and/or deletions of single characters that lead the source string to match the target string:

    a. The number of edits (insertions, substitutions, deletions) to match the source and target string is the Levenshtein Distance.
    b. This distance (d) is divided by the length of the longer string to compare (source or target).
    c. 1 – d is the Levenshtein similarity. Results closer to 1 suggest a high similarity between a pair of strings.

2. Cosine – The cosine similarity calculates the cosine of the angle between two vectors, where each vector represents one of the strings. The result is a number between 0 and 1, where numbers closer to 1 represent closer similarity between the strings provided.

3. Monge-Elkan (Cosine) – This algorithm combines the benefits of sequence-based and set-based methods and hence is less depended on the order of phrases in the strings.

If any of the three algorithms return a score above a defined threshold, the weak token lists are compared (see example 4 below). The pair is considered to be a suspected duplicate, if at least one token appears in both lists (empty token lists match with all tokens). Each matching record will be returned as Exclusivity violation in the response with "duplicate_type": "EXCLUSIVITY".

Example 3: Fuzzy name match including standardized Legal Forms in names

| LEI sent to the facility | LEIs in the Database | Potential Duplicates |
|---|---|---|
| LegalName:<br>Flux Project, LLC | LEI A: LegalName:<br>Flux Project Limited Liability Company<br><br>LEI B: LegalName:<br>Other Project, LLC<br><br>LEI C: LegalName:<br>Flux Project GmbH<br><br>LEI D: Legal Name:<br>Flex Project | **LEI A and LEI D** |

Weak tokens are words that are identified as less important for distinguishing entities than the main part of the legal name. Typically, those tokens appear in many names, have a considerable number of characters and therefore increase the similarity score to levels which reach the required threshold of a fuzzy algorithm. These string elements include standardized legal forms and typical terms with high frequency (e.g., bank, investment). In cases where the string match is identified only on weak tokens, the match is considered as a false positive and not presented to the user.

Example 4: Weak token match only

| LEI sent to the facility | LEIs in the Database | Potential Duplicates |
|---|---|---|
| LegalName:<br>AB Handelsbolag | LEI A: LegalName:<br>XY Handelsbolag<br><br>LEI B: LegalName:<br>Other Handelsbolag | **<none>** |

## 2.3 Post-processing

Post-processing is performed on all records found in the Registration Authority comparison and name matching steps in order to remove false positives from the list of potential duplicates returned to LEI Issuers. The post-processing consists of three steps: Legal Jurisdiction comparison, extended fund analysis and record deduplication.

1. Legal Jurisdiction comparison: For potential duplicates identified by the name matching algorithms, it is checked if they have the same Legal Jurisdiction. If the Legal Jurisdiction differs,

the record is not presented as a potential duplicate. Only the country-part of the Legal Jurisdiction field is compared (e.g.: if one record has Legal Jurisdiction “US-DE” and the other “US-NY”, both jurisdictions are considered to be the same). If a record does not provide a Legal Jurisdiction, it will not be filtered out based on Legal Jurisdiction.

**Example 5: Post processing based on legal jurisdiction**

| LEI sent to the facility | LEIs in the Database | Potential Duplicates |
|---|---|---|
| LegalName:<br>Power Steels Private Limited<br>Legal Jurisdiction:<br>US | LEI A: LegalName:<br>Power Steels Private Limited<br>Legal Jurisdiction:<br>IN<br><br>LEI B: LegalName:<br>Power Steels Private Limited<br>Legal Jurisdiction:<br>US-NY | **LEI B** |

2. Extended fund analysis: If one of the records has EntityCategory “FUND”, Extended Exact Prefix-Suffix match is performed to avoid multiple false hits for fund series that vary only on a number.

**Example 6: Post processing for FUNDS**

| LEI sent to the facility | LEIs in the Database | Potential Duplicates |
|---|---|---|
| LegalName:<br>Investment Fund 5<br>EntityCategory:<br><none> | LEI A: LegalName:<br>Investment Fund 4<br>EntityCategory:<br>FUND<br><br>LEI B: LegalName:<br>Investment Fund 3<br>EntityCategory:<br><none><br><br>LEI C: LegalName:<br>Umbrella Fund - Investment Fund 5<br>EntityCategory:<br>FUND | **LEI B and LEI C** |

3. Record deduplication: As potential duplicates are identified in several steps, the same record could match on different criteria in the core algorithm (e.g.: Registration Authority comparison and fuzzy string matching) and be returned to LEI Issuers multiple times. In order to reduce the effort of the LEI Issuers, Record deduplication is then applied and only one instance of the record is returned as potential duplicate.