|  |
| --- |
|  |
| GLEIF PARSER and Visualizer  High Level Design Document |
| |  |  |  | | --- | --- | --- | |  | 5 January 2018 | Version 1.0 | |

**Revision History**

**Change Record**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Author(s)** | **Version** | **Change reference** |
| 05.12.2018 | Aarthi Ilangovan  Bhuvanesh Leelakrishnan  Sowmya Prakash  Yamuna Nagasandra Rajaiah | 1.0 | Initial Version |
|  |  |  |  |
|  |  |  |  |

**Reviewers**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Version approved** | | **Date** |
| Stefan Willi Hart |  |  | |
|  |  |  | |
|  |  |  | |

Contents

[Introduction 3](#_Toc535612103)

[Purpose/Audience 3](#_Toc535612104)

[Project Requirement 3](#_Toc535612105)

[Goals and Objectives 3](#_Toc535612106)

[Functional Requirements 4](#_Toc535612107)

[Non - Functional Requirements 4](#_Toc535612108)

[Tech Stack 5](#_Toc535612109)

[Conceptual ER Diagram: Figure-1 6](#_Toc535612110)

[Description of Level 1 Diagram 7](#_Toc535612111)

[Conceptual ER Diagram: Figure-2 8](#_Toc535612112)

[Description of Level 2 RR Diagram 8](#_Toc535612113)

[Conceptual ER Diagram: Figure-3 9](#_Toc535612114)

[Description of Level 2 EXCEPTION Diagram 9](#_Toc535612115)

[Use-case Diagram 10](#_Toc535612116)

[Solution Overview 11](#_Toc535612117)

# Introduction

This document describes the high-level architecture of GLEIF Parser and Visualizer. GLEIF stands for Global Legal Entity Identifier Foundation, which manages LEI’s of all the legal entities participating in financial transactions. GLEIF stores these LEI data in various formats and one such format is XML which can be downloaded for further user processing or use. The requirement of this project is to parse the XML files which holds the Level 1 and Level 2 data of GLEIF and store it in SAP HANA database. Furthermore, reports are displayed in a web page enabling user-search facility to apply different filter criteria for viewing the GLEIF data.

# Purpose/Audience

The audience for this document is the development team and manager of this project. This document explains the architecture diagram that would be used for developing a software product. The architecture diagram provides an overview of an entire system, identifying the main components of the system and the relation with their interfaces. The project manager verifies the design to see if all the functionality is covered as part of the project.

# Project Requirement

The system expectation is to parse an GLEIF XML data source, where we have considered 3 different types of data structures available i.e., Level 1 LEI data, Level 2 Relationship Record and level 2 Reporting Exception data records for which a upload interface (web page) is provided to upload the xml file.

**Assumptions:**

1. The uploaded file format would be zip, from which xml document is extracted and converted to flattened format in SAP Hana system, to visualize the required data in the new built-in application.
2. The application doesn’t maintain any historical data when a new xml file is uploaded, the relevant table records will be deleted and parsed xml data will be loaded fresh into SAP Hana tables.
3. Extension of table relations is not taken into consideration for table storage.

# Goals and Objectives

The project focuses on the data parsing of GLEIF golden data in XML format and storage into SAP HANA database. Column-based table structure is used that is capable of better cache efficiency, since data transport between memory and CPU cache would reduce resulting in good performance and better compression rate in the system. In real time scenarios, In-memory database architecture of SAP HANA enables parallel processing of massive volume of data due to its ability to use multi-core and multi-CPU architectures to their fullest potential. GLEIF data comprises of unique constraint called Legal Entity Identifier (LEI), which are assigned for legal entities participating in financial transactions. The GLEIF data source consists of Level 1 and Level 2 Data structure. Level 1 data structure provide high level business card information and Level 2 gives overview about relationship between parent entity and child entity.

The challenge here is to efficiently parse the GLEIF data source so that quality of Legal Entity Identifier (LEI) is maintained, in case of traditional methods this process consumes more time due to large volume of data and available methods do not support efficient query processing. The data extracted from source is high in volume that is in the range of several Gigabytes (millions of records). SAP HANA database is capable of processing the query in main memory and applications that are time-sensitive would leverage the benefit.

So, an efficient mechanism is applied to handle the GLIEF data for all processing types. This data structure has been carefully examined and analysed to build application to parse the XML file and store the resulted entities into SAP HANA for faster data access. XML plays a vital role to exchange data over internet. Application requires data in XML format for further data processing. The proposed data model would give a refined idea on the workflow of implementation in terms of Data Collection, Transformation and Loading into suitable model for the benefits of the use cases. The framework is capable of providing a platform where user is intended to provide the input file and further processing is application dependent to process and analyse the data instances tagged within the file.

# Functional Requirements

* Application should be able to load GLEIF XML files from depending on user needs.
* GLIEF data source: Understanding final HANA table structure.
* Application should be able to parse XML files:
  + Automatically parse XML files into SAP HANA using JAVA.
  + XML files which is of unstructured data type. Parsing XML structure of nested fields are challenging.
* Application user interface should provide a upload button to upload GLEIF zip file from which XML will be extracted.
  + Search criteria to be Country name, File Content Date, Registration Status and Active LEI.

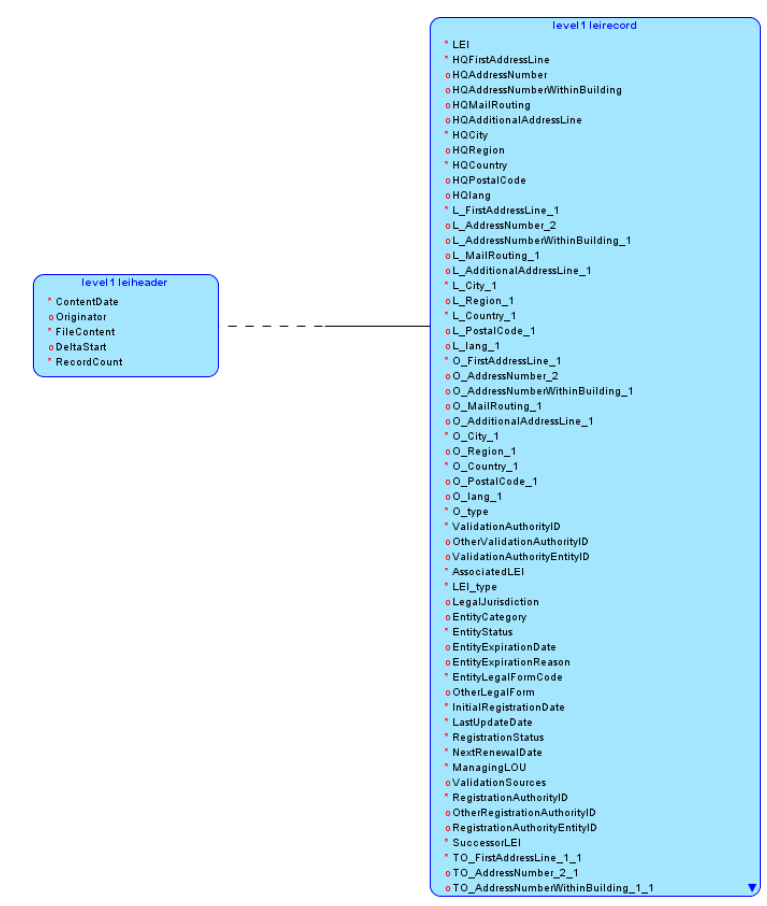
# Non - Functional Requirements

* Performance (optimum response time)
* Fault tolerance
* Portability
* Usability
* Availability
* Maintainability

# Tech Stack

|  |  |  |
| --- | --- | --- |
| **Area** | **Framework Name** | **Description** |
| J2EE/Web framework | Spring boot 2.1 (jdk 8 and above) / Spring 5.0 | Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that one can "just run". The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform. |
| Parsing XML files | JAXB/ Stax |  |
| Database interaction | JPA/Hibernate |  |
| Database | SAP HANA | Java connects to SAP HANA via ngdbc.jar |
| Frontend template engine | Thymeleaf | Thymeleaf is a Java template engine for processing and creating HTML, XML, JavaScript, CSS, and text. |
| Front End | HTML/CSS/Jquery | User needs upload button from visualization of data. |

# Conceptual ER Diagram: Figure-1



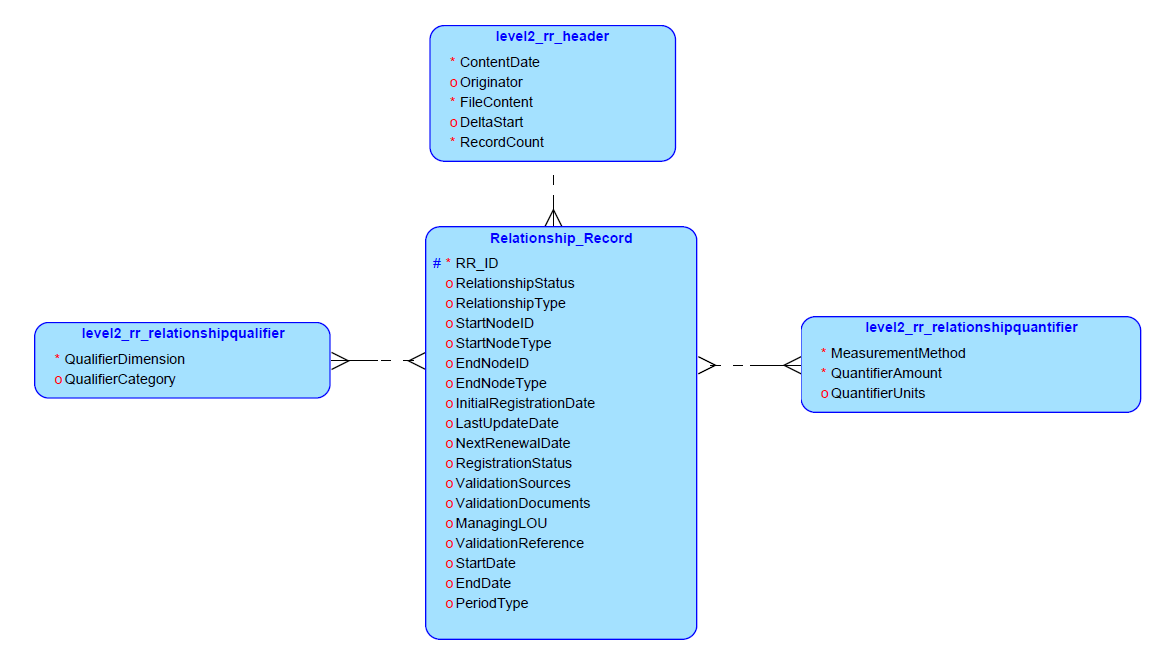
**Fig1. Level 1 Diagram**

# Description of Level 1 Diagram

This table structure contains element PNI which replaces LEI i.e., unique to PNI XML schema. A brief interpretation is mentioned below:

* level1\_leiheader contains header information with ContentDate, Originator, FileContent, DeltaStart and RecordCount accordingly.
* level1\_leirecord contains the headquarter address of parent metadata or legal entity, legal address and other address related to PNI information, Legal Jurisdiction, Entity Status, Entity Expiration Date, Entity Expiration Reason information, where date of expiry and the reason for the expiry would be recorded. Also, registration details with date of the first LEI assignment, last updated date of LEI, Registration and validation information can be found. Successor LEI, which is the LEI of the entity or entities acquires the expired entity.
* The Cardinality in most of the cases here is found to {0, 1} or {1,1} relationship, where element need not appear. It may be repeated any number of times.

# Conceptual ER Diagram: Figure-2



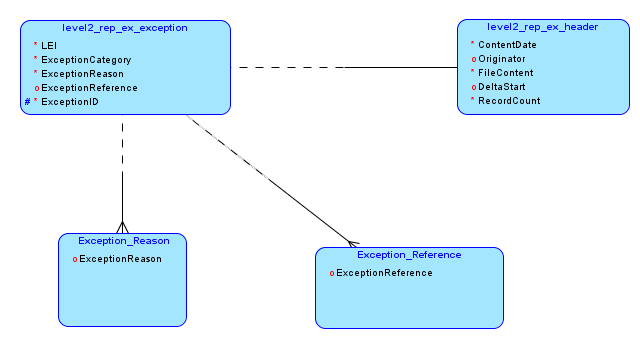
**Fig2. Level 2 RR Diagram**

# Description of Level 2 RR Diagram

This table structure contains element PNI which replaces LEI i.e., unique to PNI XML schema.

* level2\_rr\_header contains the context for the Relationship Records contained in the main body of the file. The header contains ContentDate, Originator, FileContent, DeltaStart and RecordCount based on user file input.
* Relationship\_Record inputs detailing on relationship between two legal entities if the status of relationship if active or inactive. Also, consists of registration of LEI with validation information.
* level2\_rr\_relationshipqualifier contains standards for qualifier dimension and qualifier category.
* level2\_rr\_relationshipquantifier comprises of measurement method for accounting consolidation information.
* The Cardinality in most of the cases here is found to be Mandatory, repeatable {0, \*} relationship.

# Conceptual ER Diagram: Figure-3



**Fig3. Level 2 Exception Diagram**

# Description of Level 2 Exception Diagram

This section specifies abstract data content of a GLEIF data file conforming to standards consisting of:

* level2\_rep\_ex\_header contains the context for the Relationship Records contained in the main body of the file. The header contains ContentDate, Originator, FileContent, DeltaStart and RecordCount based on user file input.
* level2\_rep\_ex\_exception has zero or more exception items and so, tables Exception\_Reason and Exception\_Reference are created, so that one of the values contained in these table are pointed out in the Exception table.
* The Cardinality of the tables are mentioned below:

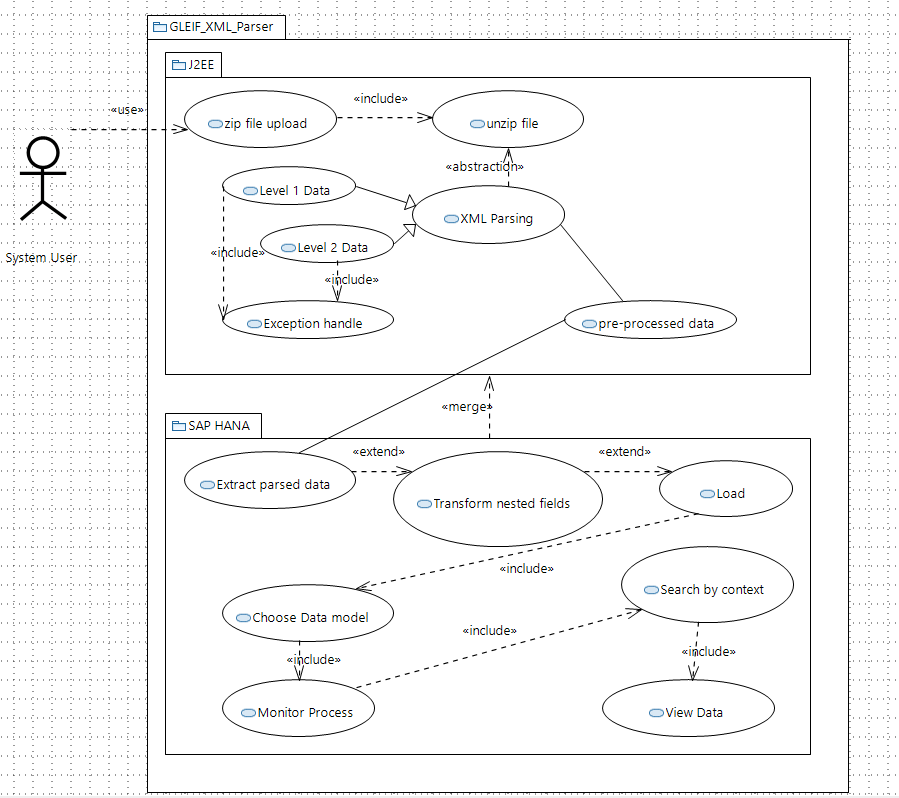
• level2\_rep\_ex\_header table is said to be Mandatory, Unique: {1,1}, which says elements must appear exactly once

• level2\_rep\_ex\_exception has Mandatory, repeatable {0, unbounded} relationship, where element need not appear. It may be repeated any number of times.

• Exception\_Reason hold Mandatory, Unique: {1,1}, which says elements must appear exactly once

• Exception\_Reference possess Mandatory, Unique: {1,1}, where elements are meant to appear exactly once.

# Use-case Diagram



# Solution Overview

