Yugandhar

Open Master Data Management (MDM) Hub

Architectural Overview

Yugandhar Open MDM Hub Release - V1.0.0
Date – 27/12/2017

Contents

1. About Yugandhar Open MDM Hub Project

Master Data Management came a long way in last decade or so. There are currently more than 20 MDM solutions catering to various specializations of MDM like Customer Data Integration (CDI), Product Information Management (PIM), vendor and supplier management etc. However most of these solutions come with licensing costs amounting to thousands of dollar. To offer a completely free solution which would be made available through Apache 2.0 license, A Project is started in 2017 under the name 'Yugandhar Open MDM Project' to build Open Source MDM solutions catering to CDI, PIM and Data Governance Capabilities. Yugandhar in Sanskrit means Ever Lasting and the strongest of its time. Our vision is to build the strongest, Open Source, Multi Domain, Cross Industry and completely free MDM Solution.

We are happy to announce that the first release of the Yugandhar MDM Hub catering to CDI solution is built with Open source technologies like Spring and Hibernate etc, inbuilt data Model, 400+ ready to use services and having incredible Out of the Box capabilities is currently being distributed. We aim to make the current CDI offering the strongest and Planning to bring Data Stewardship and PIM solutions in upcoming years.

About this document

This document provides architectural overview of Yugandhar Open MDM hub.

2. Architectural Overview of Yugandhar Open MDM Hub

Yugandhar Open Master Data Management Hub is Spring boot based Enterprise application catering to Customer Data Integration capability of MDM. It provides Golden view of the customer and accounts of your organization. It is a complete package having pre-defined data model, 450+ services which can be used to perform simple CRUD (Create, Update, retrieve, Delete) transactions as well as highly complex search and de-duplication (match and Merge) Operations.

It is designed and built using Service Oriented Architecture (SOA) having multiple loosely coupled components built on Spring Boot 1.5.6.RELEASE. It uses JavaScript Object Notation (JSON) based messages through RESTful services as well as MQ-JMS integration for exchanging the data through different system interfaces. It uses multiple open source technologies like Hibernate ORM framework for managing the persistence, ehcache for managing cache and Logback for logging.

Integration with Yugandhar Open MDM is possible though one of the below supported interfaces including

1. RESTFul webservices using json
2. JMS

However using Spring technology you can extend Yugandhar Open MDM interfaces to potentially integrate with any system which can be built in the IT landscape.

2.1 Capabilities and Silent Features

Silent Features

- Providing Industry standard MDM solution for harmonization of Customer Master Data.
- In built and Extendable Customer Master Data Model for maintaining Person and Organization master data along with Account, Address, Phone Number, Identifiers, Grouping, Vehicle, Preference, Property and party relationships etc.
- Built using Open Source technologies – Spring boot, Hibernate and JBoss
- Built on Service Oriented Architecture (SOA) having 450+ prebuilt ready to use services
- RESTful and MQ enabled
- Rapid Development using Hibernate Reverse Engineering Tool and Yugandhar Code generation Templates (Freemarker).

Capabilities

1. Free and Open Data Model
2. 450+ prebuilt ready to use services, 50+ Composite and 400 Base table services
3. Embedded and extendable data validations and business rules
4. Multilingual Reference data management Capability
5. Comprehensive application and performance logging using logback.
6. User and role based Authorization
7. Inquiry level and Pagination for retrieve and search services
8. Audit Log for maintaining database row level transaction history
9. Pluggable and UUID based Primary Key generator
10. jsr107 compliant Ehcache based caching framework
11. Integrated with Maven
12. Event Manakger
13. Match and Merge framework (Beta)

2.2 Architecture – Deep Dive

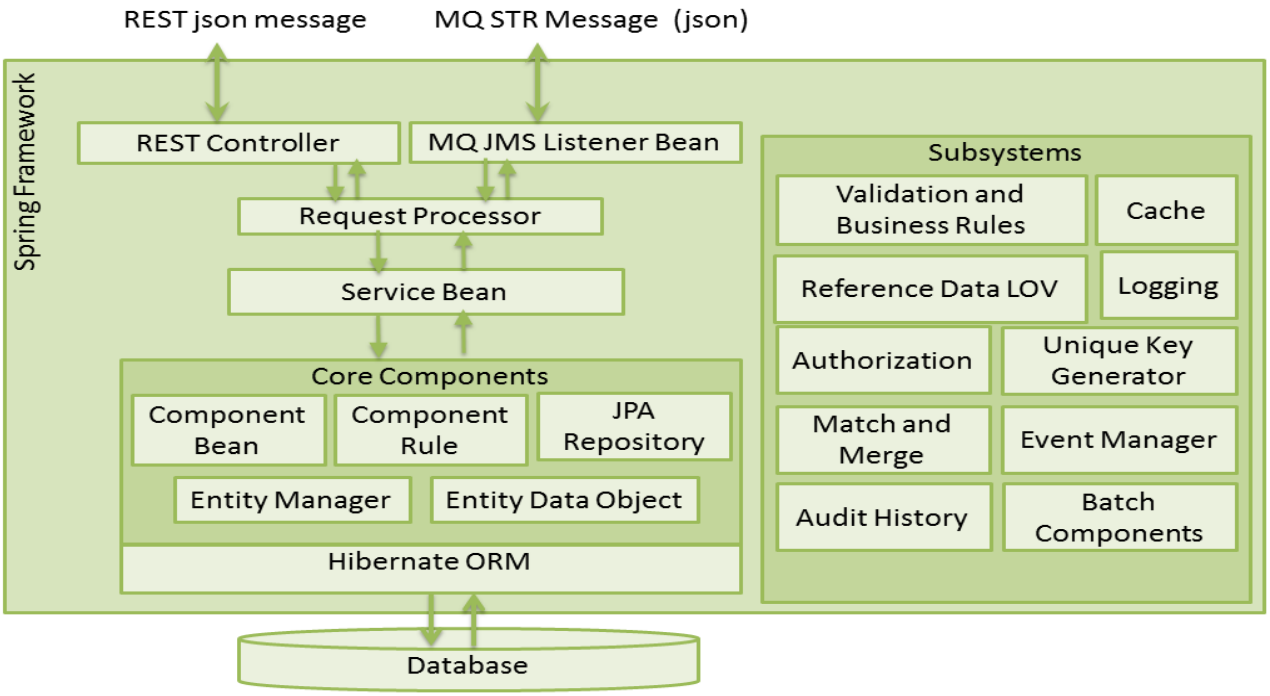

Fig. Yugandhar Open MDM Hub Application Architecture

Yugandhar Open MDM Hub is having highly simplified framework as covered in the above diagram. Let's understand the component in brief

2.2.1 Interface Components – RESTFul and JMS

2.2.1.1 REST Controller

Yugandhar Open MDM Hub provides default REST controller DefaultJsonRESTController is present in the package com.yugandhar.rest.controller. It is configured to accept json messages using POST method. This can be invoked using REST client using below link

http://<hostname>:<port>/YugandharBootProject-<SNAPSHOT version>/jsonrest/YugandharRequestProcessor

Please note – The above link is for example only. The actual link may vary if you change the request mapping of the REST controller.

After receiving the message, this controller sends the message to request processor for further processing.

2.2.1.2 JMS Listener

The JMS listener bean is located under com.yugandhar.jms package named

YugDefaultRequestQueueListener. By default this listener bean is mapped to destination="java:jboss/com/yugandhar/default/requestQueue" which is mapped to YUG.DEFAULT.REQUEST on JBoss ActiveMQ. After receiving the message, this Listener sends the message to request processor for further processing.

2.2.2 Request Processor

The request processor is Service type of Spring Bean which is single entry point of invoke transactions on the Hub server. This class invokes the transaction as mentioned in the incoming request and return the response. processMessage method processes the incoming message wherein it invokes the spring bean as per the transaction name mentioned in txnTransferObj.txnHeader.transactionServiceName attribute. Please refer the API reference guide for the structure of json transfer object and learn how to invoke transaction on Hub.

2.2.3 Core Components

The core components of the Hub include Service and Component Beans, Business Rule Class , JPA Repository class, Entity Data Objects (referred as DO) and jpa injected Entity Manager.

2.2.3.1 Service Bean

This is a Service type of Spring bean. The request from request Processor would always come to Service bean for further processing. The Service bean is a logical layer between RequestProcessor and Component bean introduced to create base or composite transaction and infuse any additional business logic so that Component beans are kept intact.

2.2.3.2 Component Bean

The Component beans handle the entity level data CRUD (Create, Update, Retrieve, Delete) operations. This bean refers and makes use of Entity Data Objects (DO), entity manager and JPA repository classes.

Component Bean primary have below methods

Persist() –This method is used to create a new record in a database entity.

merge() –This method is used to update existing record in a database entity based on primary key.

findById() –This method is used to retrieve existing record in a database based on primary key.

findByBusinessKey() – This method is used to search a database table based on business key e.g. firstname and lastName from Person table.

2.2.3.3 Entity Data Object Classes

The entity data objects are used by hibernate to map the DO to database entities. We have bifurcated the DOs in AbstractDO and Child DO wherein the Yugandhar team will extend the Abstract DOs in upcoming releases. The users of the Product must always add additional attributes in Child DO objects So that the changes of Yugandhar Team and users are kept independent. Please refer Development and Customization Guide to understand about making customization.

2.2.3.4 Optimistic Lock

To avoid concurrent update of the same record, MDM Hub use optimistic lock based on VERSION attribute of the database. MDM hub relies heavily on Hibernate Optimistic lock feature.

2.2.3.5 JPA Repository Class

The JPA repository is implementation of org.springframework.data.jpa.repository.JpaRepository interface used primarily to search the data from the single database based business keys.

2.2.3.6 ComponentRule class

The component rule class used to write the business rules for a specific entity transaction based on the method from component bean is invoked. The entity transaction can have rules. The business rules are defined at below levels based on the transaction

Rule cross points for Persist() method:

prevalidatePersit

preExecutePersist

postExecutePersit

Rule cross points for Merge() method

PrevalidateMerge

preExecuteMerge

postExecuteMerge

Rule cross points for FindById()

postExecuteFindById

prevalidateFindById

Rule cross points for FindByBusinessKey() method

postExecuteFindByBusinessKey

prevalidateFindByBusinessKey

This rule class is carved out separately so that the users make the changes in the rule class using aop instead of the Component class itself. This way any future changes in the component class made by Yugandhar Development Team will not mess up with the changes of users.

2.2.3.7 JPA injected Entity Manager

This is the default PersistenceContext used by Spring framework to make database interactions.

2.3 Yugandhar Data Model

Yugandhar Open MDM Hub comes with predefined data model to store Legal entities of type Person and Corporation master data along with its Account, Address, Phone Number, Identifiers, Grouping, Vehicle, Preference, Property and party relationships etc.

Refer the Database documentation to understand more..

2.4 Subsystems

2.4.1 Validation and Business Rules

Validation and Business rules are defined in Component rule classes. Yugandhar Open MDM Hub provides several Out of the Box (OOTB) business validations for Create, update, retrieve, find and search transactions of every entity. These rules are extendable and developer can write their own code directly in the rule class or can use Aspect Oriented programming (AOP) feature of Spring to add completely new logic or change the logic already written in OOTB code base. A sample rule override aspect YugandharRuleOverrideAspect is provided in the package com.yugandhar.common.aop.ruleoverride along with code base which have ProceedingJoinPoint on the preExecuteLegalentityCompMerge method of com.yugandhar.mdm.corecomponent.LegalentityComponentRule class.

Refer Spring Documentation for detailed understanding of AOP

https://docs.spring.io/spring/docs/4.3.x/spring-framework-reference/html/aop.html

https://docs.spring.io/spring/docs/current/spring-framework-reference/core.html

2.4.2 Cache

Yugandhar Open MDM Hub cache the Reference Data values (List of values stored as key-value pairs) so to improve transaction response time. It uses ehcahce as ehcache is an open source, standards-based cache that boosts performance, offloads your database, and simplifies scalability. It's the most widely-used Java-based cache because it's robust, proven, full-featured, and integrates with other popular libraries and frameworks. The ehcache configuration file is kept in /src/main/resources/ehcache.xml of the YugandharBootProject of the code base. The default ehcache expiry time is configured as 1296000 seconds (15 days) in the ehcache.xml configuration file. This may be revisited as per the requirements. Visit www.ehcache.org for understanding ehcache in further detail.

2.4.3 Reference Data (LOV) Management

Yugandhar Open MDM Hub supports storing the reference data Management and have standard pre-defined format for storing the key-value pairs. This format is extendable as per requirements. The typical Reference data table starts with the name REF_ e.g. REF_COUNTRY_ISO which is a LOV which stores the ISO Country Codes which can be used in storing address or country of domicile etc. The reference data table has CONFIG_LANGUAGE_CODE_KEY, KEY, VALUE and DESCRIPTION attributes as predefined format. The CONFIG_LANGUAGE_CODE_KEY attribute provides much needed multilingual support.

e.g. The country 'France' in English is written as फ्रांस in hindi and as 'francés' in Spanish. So the LOV can be populated as

CONFIG_LANGUAGE_CODE_KEY	KEY	VALUE	Description
1 (for English)	250	FRA	France
2 (For Hindi)	250	FRA	फ्रांस
3 (For Spanish)	250	FRA	'francés'

This way, based on the requestor Language of the transaction the relevant record can be fetched from the database and returned in the response.

2.4.4 Authorization Framework

Yugandhar Open MDM Hub provides user and role based Authorization framework. The tables AUTH_ROLES_REGISTRY, AUTH_USER_REGISTRY, AUTH_USER_ROLE_ASSOC and AUTH_USERROLE_ACCESSCONTROL comprise of the Authorization framework.

AUTH_ROLES_REGISTRY – stores the list of roles

AUTH_USER_REGISTRY – stores the list of users

AUTH_USER_ROLE_ASSOC – stores the user to role mapping

AUTH_USERROLE_ACCESSCONTROL – Stores if the user or role is authorized to perform given transaction. List of transaction is refered from CONFIG_TXN_REGISTRY table and the ID_PK of CONFIG_TXN_REGISTRY table is referred as FK to provide the access.

Refer Database documentation for further understanding of the database. Also refer the Authorization framework section of the API reference guide to understand the OOTB transaction used to make entries these tables.

2.4.5 Unique key (Primary key) Generator

Yugandhar Open MDM Hub currently uses UUID based Primary key generator. This implementation class of the yugandhar key generator is AbstractYugandharKeygenerator. However if you want to customize the Unique key generation logic then you may write your custom logic in the below class

com.yugandhar.mdm.keygen.YugandharKeygenerator

The Primary key generation logic is externalized using YugandharKeygenerator class mentioned above. Please make sure that you should not modify the AbstractYugandharKeygenerator as it is supposed to be modified only by Yugandhar development team.

2.4.6 Logging

Yugandhar Open MDM Hub have comprehensive Logging framework for application logging. It uses logback as the logging framework as logback is much more advanced that the log4j logging. The logback configuration file is kept in /YugandharBootProject/src/main/resources/yugandhar_logback.xml location in the code base. By default the SizeAndTimeBasedFNATP rolling file appeder is used for the logging but this can be modified as per requirements. Please refer below url to learn more about logback

https://logback.qos.ch/

2.4.7 Audit History

Yugandhar Open MDM Hub have a comprehensive data model to store the Audit logic. Each entity table have Audit Log table appended with 'AL_' which stores all the Insert, Update and Delete related changes to the base table. One additional row gets created in 'AL_' tables after every Insert, Update and Delete on the record of the base table so that the complete history of changes on the data record gets tracked.

2.4.8 Batch Processing Server Component

At present, the Yugandhar MDM provides basic data model to mark the entities for batch processing and track the actions. An entry is made in BATCH_ENTITY_TO_PROCESS table along with Batch action and batch action status (REF_BATCH_ACTION_STATUS and REF_BATCH_PROPOSED_ACTION tables) so that the entity gets identified for processing.

Please note it's considered that a database query, ETL job or any batch client will use the data from this table and invoke transaction on MDM Hub to process the entities. The Batch processor client is for Yugandhar Open MDM Hub is still not released.

2.4.9 Match and Merge Framework (Beta release)

Yugandhar Open MDM hub has comprehensive Match and Merge framework. Its currently available as a Beta release. It supports Deterministic and Fuzzy matching

Deterministic matching – The data values of attributes are considered equal if exact match.

Fuzzy matching – The match candidates are searched based on phonetic match of the attributes and considered match if the values are equal.

The Data Model -

- **MATCH_CANDIDATE_LE_REGISTRY -**This table stores the match results after performing matching on perticular legal entity. e.g. if legal entity A1 is being matched and identified to have A2 and A3 as candidate legal entities having match pattern with A2 as YYYNYY and with A3 as NYYYYN, identified to be manual reviewed before merging then two records would be created in this table with ID_PK of A1 legalentity to be mapped to LEGALENTITY_IDPK

attribute and ID_PK of A2 being mapped to CANDIDATE_LEGALENTITYIDPK attribute. YYYNYY being mapped to MATCH_PATTERN, MATCH_PROPOSED_ACTION_REFKEY as defined in REF_MATCH_PROPOSED_ACTION, MATCH_ACTIONSTATUS_REFKEY mapped as defined in REF_MATCH_ACTIONSTATUS tables. The percentage match after performing the matching is stored inMATCH_PERCENTAGE_DESCRIPTION attribute.

- **MATCH_MERGED_LE_ASSOC -**This table stores the legal entity relation after performing the matching. e.g. if Legalentity A1 is merged with A2 where A1 is survivor then the ID_PK of A1 legal entity should be stored in SURVIVOR_LEGALENTITY_IDPK and ID_PK of A2 legal entity should be stored in MERGED_LEGALENTITY_IDPK attribute. The reason for the merge should be stored in MERGE_REASON_REFKEY attribute along with any comments in the COMMENTS column
- **REF_MATCH_ACTIONSTATUS -**The match action status LOV
- **REF_MATCH_PROPOSED_ACTION -**The match Proposed action LOV
- **REF_MATCH_RESULT -**The match result LOV used to store if the match is exact match, close match etc
- **REF_MATCH_SCORE -**The match score LOV used to store the match attribute pattern
- **REF_MATCH_THRESHOLD -** This LOV stores the threshold of the match e.g. if an attribute mathc 80% then only it should be considered a close match etc.

Java OOTB Rules –

The OOTB rules are present in the com.yugandhar.mdm.match.rules package

- LeAddressAndPhoneMatchRule
- LeCorporationDeterministicMatchRule
- LeCorporationFuzzyMatchRule
- LePersonDeterministicMatchRule
- LePersonFuzzyMatchRule
- YugandharMatchingConstants
- YugandharMatchingUtils