**Yugandhar**

**Open Master Data Management (MDM) Hub**

**Data Model Guide**

Yugandhar Open MDM Hub Release - V1.0.0

Date – 27/12/2017

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Copyright [2017] [Yugandhar Open MDM Hub]

Licensed under the Apache License, Version 2.0 (the "License");

you may not use this file except in compliance with the License.

You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and limitations under the License.

Contents

[Understanding Data Table structure 4](#_Toc502597701)

[Understanding Reference Table structure 5](#_Toc502597702)

[Understanding Configuration tables 6](#_Toc502597703)

[Understanding Audit Log table structure 7](#_Toc502597704)

[Understanding Application Configuration entities 8](#_Toc502597705)

[CONFIG\_APP\_PROPERTIES 8](#_Toc502597706)

[CONFIG\_ERRORCODE\_REGISTRY 8](#_Toc502597707)

[CONFIG\_INQUIRY\_LEVELS 8](#_Toc502597708)

[CONFIG\_LANGUAGE\_CODE 8](#_Toc502597709)

[CONFIG\_TXN\_REGISTRY 8](#_Toc502597710)

[Understanding Data Entities 9](#_Toc502597711)

[Legal entity 9](#_Toc502597712)

[Uses 9](#_Toc502597713)

[Database entities 9](#_Toc502597714)

[Legal entity - Person 10](#_Toc502597715)

[Uses 10](#_Toc502597716)

[Database 10](#_Toc502597717)

[Legal entity - Corporation 10](#_Toc502597718)

[Uses 10](#_Toc502597719)

[Database 10](#_Toc502597720)

[Legal entity - Address 10](#_Toc502597721)

[Uses 10](#_Toc502597722)

[Database 10](#_Toc502597723)

[Legal entity - Phone 10](#_Toc502597724)

[Uses 10](#_Toc502597725)

[Database 10](#_Toc502597726)

[Account 11](#_Toc502597727)

[Uses 11](#_Toc502597728)

[Database 11](#_Toc502597729)

[Account Phone 11](#_Toc502597730)

[Uses 11](#_Toc502597731)

[Database 11](#_Toc502597732)

[Account Address 11](#_Toc502597733)

[Uses 11](#_Toc502597734)

[Database 11](#_Toc502597735)

[Entity Groups 11](#_Toc502597736)

[Uses 11](#_Toc502597737)

[Database 11](#_Toc502597738)

[LE Property 12](#_Toc502597739)

[Uses 12](#_Toc502597740)

[Database entities 12](#_Toc502597741)

[LE Vehicle 13](#_Toc502597742)

[Uses 13](#_Toc502597743)

[Database 13](#_Toc502597744)

[Identifier 13](#_Toc502597745)

[Uses 13](#_Toc502597746)

[Database 13](#_Toc502597747)

[MISCELLANEOUS\_INFO table 13](#_Toc502597748)

[Uses 13](#_Toc502597749)

[Database entities - 13](#_Toc502597750)

[List of Reference Data (LOV) entities 14](#_Toc502597751)

[List ofAudit History Logging entities 16](#_Toc502597752)

# 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 covers the data model of Yugandhar Open MDM Hub.

# Understanding Data Table structure

The data entity table has below attributes in common. All the other attributes are used to store the data related to the entity however below attributes are mandatory for Yugandhar Code generation framework to work.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Primary key | Nullable? | Data Type | Description |
| ID\_PK | YES | No | VARCHAR2 (50 Byte) | ID to uniquely identify an entity in the system |
| VERSION |  | No | NUMBER | VERSION attribute used for optimistic lock |
| CREATED\_TS |  | No | TIMESTAMP(6) | Creation timestamp of the record |
| DELETED\_TS |  | YES | TIMESTAMP(6) | Soft-delete timestamp of the record |
| UPDATED\_TS |  | No | TIMESTAMP(6) | The timestamp when the record is last updated |
| UPDATED\_BY\_USER |  | No | VARCHAR2 (50 Byte) | The userid which updated this record last |
| UPDATED\_BY\_TXN\_ID |  | YES | VARCHAR2 (100 Byte) | The reference id of the transaction which updated the record |

# Understanding Reference Table structure

The reference data entity tables are meant to store key-value pairs. This is also called termed as ‘List of Values’ (LOB). LOV tables have some soft rules to easily identify them from other tables as below

1. The Name of reference data entity must start with ‘REF\_’ characters
2. The CONFIG\_LANGUAGE\_CODE\_KEY attribute must be present for multilingual support
3. The entity must have KEY and VALUE column.
4. By default, Yugandhar Open MDM Hub Reference data entity generated transactions does the search and matching on KEY attribute. The same should be followed going forward. However if needed the code can be modified to do the searching based on VALUE column as well.
5. The CONFIG\_LANGUAGE\_CODE\_KEY and KEY attribute forms the unique constraint to restrict creating any duplicate values in the reference list.
6. The records of the reference data entities get cached in Yugandhar Open MDM Hub using ehcache caching framework. This is done to improve the performance of real time transactions. You have the option to disable the caching either by removing the cache statement or changing the cache expiry time to few milliseconds. However it’s not advisable to disable the caching unless there is a strong reason to disable the same.

The reference data entity tables have below attributes in common.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Primary key | Nullable? | Data Type | Description |
| ID\_PK | Yes | N | VARCHAR2 (50 Byte) | ID to uniquely identify an entity in the system |
| VERSION |  | N | NUMBER | VERSION attribute used for optimistic lock |
| CREATED\_TS |  | N | TIMESTAMP(6) | Creation timestamp of the record |
| DELETED\_TS |  | Y | TIMESTAMP(6) | Soft-delete timestamp of the record |
| UPDATED\_TS |  | N | TIMESTAMP(6) | The timestamp when the record is last updated |
| UPDATED\_BY\_USER |  | N | VARCHAR2 (50 Byte) | The userid which updated this record last |
| UPDATED\_BY\_TXN\_ID |  | Y | VARCHAR2 (100 Byte) | The reference id of the transaction which updated the record |
| CONFIG\_LANGUAGE\_CODE\_KEY | Unique Key | N | VARCHAR2 (50 Byte) | The language code which provide multilingual support. This language code is refered from application configuration table CONFIG\_LANGUAGE\_CODE. Not to be confused with REF\_LANGUAGE\_CODE, which gets used to store the list of languages to be used in conjunction with data like preferred language of the person. |
| KEY | Unique Key | N | VARCHAR2 (50 Byte) | The key of the key-value pair from list of values |
| VALUE |  | N | VARCHAR2 (50 Byte) | The value of the key-value pair from list of values |
| DESCRIPTION |  | Y | VARCHAR2 (50 Byte) | Description if any related to key value pair |

# Understanding Configuration tables

The configuration tables starts with prefix ‘CONFIG\_<entityname>’. The structure of the entity is similar to the data entities mentioned above. Also most of the configuration entities are cached, yet there is no fixed guideline on the same. The configuration tables are exclusively used by Yugandhar Framework and must not be customized by the user.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Primary key | Nullable? | Data Type | Description |
| ID\_PK | YES | No | VARCHAR2 (50 Byte) | ID to uniquely identify an entity in the system |
| VERSION |  | No | NUMBER | VERSION attribute used for optimistic lock |
| CREATED\_TS |  | No | TIMESTAMP(6) | Creation timestamp of the record |
| DELETED\_TS |  | YES | TIMESTAMP(6) | Soft-delete timestamp of the record |
| UPDATED\_TS |  | No | TIMESTAMP(6) | The timestamp when the record is last updated |
| UPDATED\_BY\_USER |  | No | VARCHAR2 (50 Byte) | The userid which updated this record last |
| UPDATED\_BY\_TXN\_ID |  | YES | VARCHAR2 (100 Byte) | The reference id of the transaction which updated the record |

# Understanding Audit Log table structure

The Audit Log tables store the history Insert, update and delete operations performed on the base database entity. Audit log tables have some soft rules to easily identify them from other tables as below

1. The name of the Audit Log table starts with ‘AL\_<Name of the base entity>’ e.g. the Audit table of the LEGALENTITY table is AL\_LEGALENTITY. The name can be trimmed or changed if needed.
2. The Audit log table must have below mentioned attributes along with all the attributes of the base entity.
3. The AUDITLOG\_ACTION\_CODE gets updated as per below logic
   1. If a new record is created in base entity (e.g. LEGALENTITY) then audit log trigger will create a new record in corresponding audit log table (e.g. AL\_ LEGALENTITY) with AUDITLOG\_ACTION\_CODE as ‘I’.
   2. If a new record is updated in base entity (e.g. LEGALENTITY) then audit log trigger will create a new record in corresponding audit log table (e.g. AL\_ LEGALENTITY) with AUDITLOG\_ACTION\_CODE as ‘U’.
   3. If a new record is created in base entity (e.g. LEGALENTITY) then audit log trigger will create a new record in corresponding audit log table (e.g. AL\_ LEGALENTITY) with AUDITLOG\_ACTION\_CODE as ‘D’.
4. So Audit log tables will have multiple rows for each operation performed on every record giving a full audit history.
5. Please note that currently the Audit Log tables does not have any purging policy provided as the purging requirements are highly dissimilar so your Audit log data size may continue to rise. It is advisable to disable the audit history (by disabling relevant database triggers) if audit information is not needed or define a policy to purge in MDM project design phase itself.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Primary key | Nullable? | Data Type | Description |
| AUDITLOG\_ID\_PK | YES | N | VARCHAR2 (50 Byte) | ID to uniquely identify an entity in the system |
| AUDITLOG\_CREATED\_TS |  | N | TIMESTAMP(6) | The timestamp when audit history record is created |
| AUDITLOG\_ACTION\_CODE |  | N | CHAR (1 Byte) | The action code of the audit log record based on base table |

# Understanding Application Configuration entities

CONFIG\_APP\_PROPERTIES - The table stores the application properties, should not be changed unless needed

CONFIG\_ERRORCODE\_REGISTRY –

This table stores all the error code and error messages being referred in MDM application

CONFIG\_INQUIRY\_LEVELS –

The table is used to configure the inquiry level related to any transaction.

CONFIG\_LANGUAGE\_CODE

Configuration Language Codes supported by Application

CONFIG\_TXN\_REGISTRY

The table is reference list storing the list of all the services available for Yugandhar MDM

# Understanding Data Entities

## Legal entity

Uses – Legal entity Modelused to store the legal entities basic information. A Legal entity can be of Person (Individual) or Corporation (Organization) type. Based on the type of the legal entity the ENTITY\_OBJECT\_TYPE\_REFKEY should be populated. There can only be an entry in Person or Corporation type of a given legal entity. i.e. Person and Corporation must never have same legal entity Idpk.

Database entities -

* LEGALENTITY
* LE\_VEHICLE\_ASSOC
* LE\_TO\_LE\_RELATIONSHIP
* LE\_SYSTEM\_KEYS\_REGISTRY
* LE\_PROPERTY\_ASSOC
* LE\_PREFERENCES
* LE\_PHONE\_ASSOC
* LE\_PERSON
* LE\_IDENTIFIER\_KYC\_REGISTRY
* LE\_CORPORATION
* LE\_ADDRESS\_ASSOC
* LE\_ACCOUNT\_ASSOC

## Legal entity - Person

Uses – LE Person tables are used to store the person data

Database entities -

* LE\_PERSON
* PERSONNAMES
* REF\_PERSON\_TYPE
* REF\_PERSONNAME\_TYPE

## Legal entity - Corporation

Uses – LE Corporation tables are used to store the Corporation data

Database entities -

* CORPORATIONNAMES
* LE\_CORPORATION
* REF\_CORPORATION\_NAME\_TYPE
* REF\_CORPORATION\_TYPE

## Legal entity - Address

Uses – LE Address tables are used to store the Address data for Person and Corporation type of entities.

Database entities -

* ADDRESS
* LE\_ADDRESS\_ASSOC
* REF\_ADDRESS\_SUBTYPE
* REF\_ADDRESS\_TYPE

## Legal entity - Phone

Uses – Le Phone Model is used to store the phone, mobile, fax etc numbers of the Legal entity

Database entities -

* LE\_PHONE\_ASSOC
* PHONE\_STANDARDIZED
* REF\_PHONE\_SUBTYPE
* REF\_PHONE\_TYPE

## Account

Uses – Account Model is used to store Accounts of the Legal entity

Database entities - Tables of the Account Model include below database entities

* ACCOUNT
* ACCOUNT\_ADDRESS\_ASSOC
* ACCOUNT\_PHONE\_ASSOC
* LE\_ACCOUNT\_ASSOC
* REF\_ACCOUNT\_MDM\_STATUS
* REF\_ACCOUNT\_SOURCE\_STATUS

## Account Phone

Uses – Account phone model is used to store the phone, mobile, fax etc numbers related to account.

Database entities -

* ACCOUNT\_PHONE\_ASSOC
* PHONE\_STANDARDIZED
* REF\_PHONE\_SUBTYPE
* REF\_PHONE\_TYPE

## Account Address

Uses – Account Address model is used to store the Address related to Account.

Database Entities -

* ACCOUNT\_ADDRESS\_ASSOC
* ADDRESS
* REF\_ADDRESS\_SUBTYPE
* REF\_ADDRESS\_TYPE

## Entity Groups

Uses – Entity group’s model is used to store the grouping of the entities

Database entities -

* ENTITY\_GROUP
* ENTITY\_GROUP\_ASSOC
* REF\_GROUP\_SUBTYPE
* REF\_GROUP\_TYPE

## LE Property

Uses – This model is used to store the property details related to entities

## Database entities

* LE\_PROPERTY\_ASSOC
* PROPERTY
* REF\_PROPERTY\_LE\_RELTYPE

## LE Vehicle

Uses – This model is used to store the vehicle information related to legal entities.

Database entities -

* LE\_VEHICLE\_ASSOC
* VEHICLE

## Identifier

Uses – This model is used to store the Identification number of the legal entities

Database entities -

* LE\_IDENTIFIER\_KYC\_REGISTRY
* REF\_IDENTIFICATION\_TYPE

## MISCELLANEOUS\_INFO table

Uses – Miscellanious information is stored in this table.

## Database entities -

MISCELLANEOUS\_INFO

# List of Reference Data (LOV) entities

* REF\_ACCOUNT\_MDM\_STATUS
* REF\_ACCOUNT\_SOURCE\_STATUS
* REF\_ADDRESS\_SUBTYPE
* REF\_ADDRESS\_TYPE
* REF\_AGREEMENT\_TYPE
* REF\_ASSOC\_TYPE
* REF\_BATCH\_ACTION\_STATUS
* REF\_BATCH\_PROPOSED\_ACTION
* REF\_BILLING\_MODE\_TYPE
* REF\_BRANCH\_CODE
* REF\_CLASSIFICATION\_CODE
* REF\_CORPORATION\_NAME\_TYPE
* REF\_CORPORATION\_TYPE
* REF\_COUNTRY\_ISO
* REF\_CURRENCY
* REF\_DEACTIVATION\_REASON
* REF\_ENTITY\_OBJECT\_TYPE
* REF\_GENDER
* REF\_GROUP\_SUBTYPE
* REF\_GROUP\_TYPE
* REF\_HIGHEST\_EDU\_QUAL
* REF\_IDENTIFICATION\_TYPE
* REF\_IMPORTANCE\_TYPE
* REF\_INACTIVATION\_REASON
* REF\_INDUSTRY\_CODE
* REF\_LANGUAGE\_CODE
* REF\_LE\_RATING
* REF\_LE\_RELATIONSHIP\_TYPE
* REF\_LE\_ROLETYPE
* REF\_LOBTYPE
* REF\_MATCH\_ACTIONSTATUS
* REF\_MATCH\_PROPOSED\_ACTION
* REF\_MATCH\_RESULT
* REF\_MATCH\_SCORE
* REF\_MATCH\_THRESHOLD
* REF\_MERGE\_REASON
* REF\_PERSON\_TYPE
* REF\_PERSONNAME\_TYPE
* REF\_PHONE\_SUBTYPE
* REF\_PHONE\_TYPE
* REF\_PREFERENCE\_TYPE
* REF\_PREFIX\_NAME
* REF\_PROPERTY\_LE\_RELTYPE
* REF\_RELATIONSHIP\_STATUS
* REF\_SOURCE\_SYSTEM
* REF\_STATE\_PROVINCE
* REF\_STATUS\_IN\_SOURCE
* REF\_STATUS\_TYPE
* REF\_SUFFIX\_NAME
* REF\_TERMINATION\_REASON

# List ofAudit History Logging entities

* AL\_ACCOUNT
* AL\_ACCOUNT\_ADDRESS\_ASSOC
* AL\_ACCOUNT\_PHONE\_ASSOC
* AL\_ADDRESS
* AL\_AUTH\_ROLES\_REGISTRY
* AL\_AUTH\_USER\_REGISTRY
* AL\_AUTH\_USER\_ROLE\_ASSOC
* AL\_AUTH\_USERROLE\_ACCESSCONTROL
* AL\_BATCH\_ENTITY\_TO\_PROCESS
* AL\_CONFIG\_APP\_PROPERTIES
* AL\_CONFIG\_ERRORCODE\_REGISTRY
* AL\_CONFIG\_INQUIRY\_LEVELS
* AL\_CONFIG\_LANGUAGE\_CODE
* AL\_CONFIG\_TXN\_REGISTRY
* AL\_CORPORATIONNAMES
* AL\_ENTITY\_GROUP
* AL\_ENTITY\_GROUP\_ASSOC
* AL\_INACTIVE\_LE\_REGISTRY
* AL\_LE\_ACCOUNT\_ASSOC
* AL\_LE\_ADDRESS\_ASSOC
* AL\_LE\_CORPORATION
* AL\_LE\_IDENTIFIER\_KYC\_REGISTRY
* AL\_LE\_PERSON
* AL\_LE\_PHONE\_ASSOC
* AL\_LE\_PREFERENCES
* AL\_LE\_PROPERTY\_ASSOC
* AL\_LE\_SYSTEM\_KEYS\_REGISTRY
* AL\_LE\_TO\_LE\_RELATIONSHIP
* AL\_LE\_VEHICLE\_ASSOC
* AL\_LEGALENTITY
* AL\_MATCH\_CANDIDATE\_LE\_REGISTRY
* AL\_MATCH\_MERGED\_LE\_ASSOC
* AL\_MISCELLANEOUS\_INFO
* AL\_PERSONNAMES
* AL\_PHONE\_STANDARDIZED
* AL\_PROPERTY
* AL\_REF\_ACCOUNT\_MDM\_STATUS
* AL\_REF\_ACCOUNT\_SOURCE\_STATUS
* AL\_REF\_ADDRESS\_SUBTYPE
* AL\_REF\_ADDRESS\_TYPE
* AL\_REF\_AGREEMENT\_TYPE
* AL\_REF\_ASSOC\_TYPE
* AL\_REF\_BATCH\_ACTION\_STATUS
* AL\_REF\_BATCH\_PROPOSED\_ACTION
* AL\_REF\_BILLING\_MODE\_TYPE
* AL\_REF\_BRANCH\_CODE
* AL\_REF\_CLASSIFICATION\_CODE
* AL\_REF\_CORPORATION\_NAME\_TYP
* AL\_REF\_CORPORATION\_NAME\_TYPE
* AL\_REF\_CORPORATION\_TYPE
* AL\_REF\_COUNTRY\_ISO
* AL\_REF\_CURRENCY
* AL\_REF\_DEACTIVATION\_REASON
* AL\_REF\_ENTITY\_OBJECT\_TYPE
* AL\_REF\_GENDER
* AL\_REF\_GROUP\_SUBTYPE
* AL\_REF\_GROUP\_TYPE
* AL\_REF\_HIGHEST\_EDU\_QUAL
* AL\_REF\_IDENTIFICATION\_TYPE
* AL\_REF\_IMPORTANCE\_TYPE
* AL\_REF\_INACTIVATION\_REASON
* AL\_REF\_INDUSTRY\_CODE
* AL\_REF\_LANGUAGE\_CODE
* AL\_REF\_LE\_RATING
* AL\_REF\_LE\_RELATIONSHIP\_TYPE
* AL\_REF\_LE\_ROLETYPE
* AL\_REF\_LOBTYPE
* AL\_REF\_MATCH\_ACTIONSTATUS
* AL\_REF\_MATCH\_PROPOSED\_ACTION
* AL\_REF\_MATCH\_RESULT
* AL\_REF\_MATCH\_SCORE
* AL\_REF\_MATCH\_THRESHOLD
* AL\_REF\_MERGE\_REASON
* AL\_REF\_PERSON\_TYPE
* AL\_REF\_PERSONNAME\_TYPE
* AL\_REF\_PHONE\_SUBTYPE
* AL\_REF\_PHONE\_TYPE
* AL\_REF\_PREFERENCE\_TYPE
* AL\_REF\_PREFIX\_NAME
* AL\_REF\_PROPERTY\_LE\_RELTYPE
* AL\_REF\_RELATIONSHIP\_STATUS
* AL\_REF\_SOURCE\_SYSTEM
* AL\_REF\_STATE\_PROVINCE
* AL\_REF\_STATUS\_IN\_SOURCE
* AL\_REF\_STATUS\_TYPE
* AL\_REF\_SUFFIX\_NAME
* AL\_REF\_TERMINATION\_REASON
* AL\_VEHICLE