|  |
| --- |
| Yodlee Inc. |
| Yodlee SDM Guidelines |
| How to create Yodlee SDM for YSL |
|  |
| **Hema Ramakrishnappa** |
| **3/31/2014** |

|  |
| --- |
| This document explains the structure of the Yodlee Service Data Model used by the Yodlee Service Language (also known as the Dynamic API or DAPI). It also outlines the rules and guidelines to be followed while enhancing or creating new elements in the SDM. |

Contents

[Components of the YSDM 2](#_Toc384912738)

[Entities 2](#_Toc384912739)

[Components 2](#_Toc384912740)

[Enumerations 2](#_Toc384912741)

[File Structure 2](#_Toc384912742)

[Entity Definition Format 3](#_Toc384912743)

[Enumeration Definition Format 8](#_Toc384912744)

[Understanding columns in the SDM 9](#_Toc384912745)

[1. Entity 9](#_Toc384912746)

[2. Attribute 9](#_Toc384912747)

[3. Qualifier/Container 9](#_Toc384912748)

[4. Reference 9](#_Toc384912749)

[5. DOM 9](#_Toc384912750)

[6. Type 9](#_Toc384912751)

[7. Description 9](#_Toc384912752)

[8. Value 9](#_Toc384912753)

[9. Editable 9](#_Toc384912754)

[10. Repeats 9](#_Toc384912755)

[11. Validation 9](#_Toc384912756)

[12. Dependency 9](#_Toc384912757)

[Adding Components <rename “Yodlee Data Types”> 9](#_Toc384912758)

[Adding Enumerations 10](#_Toc384912759)

[Other considerations for populating the SDM 10](#_Toc384912760)

[1. Specifying filters 10](#_Toc384912761)

[2. Specifying component types 10](#_Toc384912762)

[3. Entity/attribute names 10](#_Toc384912763)

[4. Review for correctness 10](#_Toc384912764)

[Backward Compatibility Guidelines 11](#_Toc384912765)

**Purpose of YSDM**

The YSDM is a collection of documents describing Yodlee Service Domain Model that can be accessed using Yodlee Service Language (YSL). YSDM aims to specify functional, security, performance, and customization (co-branding) aspects of service domain model. YSDM is the logical view of domain model and doesn’t match with physical DB schema and hence may contain synthetic fields and/or entities that don’t have a direct mapping to any of persistent entities/attributes.

The SDM is used internally by the DAPI engine to create rules that the engine uses to determine what data can be fetched/updated.

A subset of the SDM will be made available to external developers using the Yodlee API as part of documentation describing what data can be accessed using the API.

# Components of the YSDM

## Entities

A Yodlee object exposed via the Yodlee DAPI. Examples are accounts, goals, user profile, datapoints.

## Components

Commonly used Yodlee types. Examples are date and money.

## Enumerations

Yodlee attributes whose values belong to an enumeration are stored here.

# File Structure

The Yodlee SDM is split over multiple Excel spreadsheets to make it manageable and easy to maintain.

The SDM is located here - //razor/core/newarch/modules/sdm

Currently, entities accounts, transactions, and statements, though related, are stored in multiple files due to their large size. To add support for new containers for these entities, enhance the same files.

Unrelated entities like user profile are maintained in a separate file. When a new entity is added, use your discretion (based on its size and relationship with other entities) to determine if it fits into an existing SDM file or deserves a separate file.

# Entity Definition Format

The following table describes the format of the worksheet(s) defining (non-enumeration) entities.

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Description** | **Examples** | **Notes** |
| Entity | Name of the entity | Account, Transaction. | Logical name of YSDM entities don’t need to match entity names in YDOM. |
| Attribute | Name of attribute | Id, balance | Logical names don’t need to match names in YDOM |
| Container  /  Qualifier | **Container**: This column is applicable to account, transaction, and statement models. These entities are polymorphic and value in this column denotes whether the attribute is available to specific containers. If nothing is specified in this column, the attribute is not polymorphic.  **Qualifier**: This column can be used to denote polymorphic attributes in general.  **For entities**: The value for entity for this column denotes the enumeration that should be used as a differentiator for this entity.  **For attributes**: Leave the column blank if the attribute is not polymorphic. Otherwise, specify the value of differentiator.  One can also specify values of default enumerated filters applicable to the entity. | **Container**: clearedBalance only applies to bank container. Hence, clearedBalance will have “bank” in container column.  **Qualifier**:  Example 1: For Holding, holdingType can be used a differentiator as Holding can have different attributes based on its holdingType.  Attribute mutualFundType is only available to Holdings that have holdingType as mutualFund.  Example 2: Ownership filter applies to accounts. Ownership value shared can be used in this column to specify whether an attribute is only available in case of shared account. Example: sharedByMember. |  |
| Reference | This column is only applicable to attributes and should be used when the attribute is defined in terms of other attributes defined in the YSDM. The column can be either empty or have one of following three values:   1. Name of any other attribute 2. **Priority**: Can contain name of two attributes separated by semicolon. ATTR1;ATTR2. This will be evaluated as ATTR1 == null ? ATTR1 : ATTR2. 3. **Condition**: Boolean Expression ? Value1 : Value2   Boolean expression can be true|false, can refer to any boolean attribute, or can specify any enumerated filter value for the entity. Value1 and Value2 refer to other attributes. Minus sign (-) can also be prefixed when referring to numeric attributes. | **Reference**: Attribute “balance” in account entity is defined as another attribute “clearedBalance” for bank container.  **Priority:**  Date attribute of transaction can be defined as “postDate;TransDate” which will use postdate if available, and transDate otherwise.  **Condition**: networth of an account can be defined using balance and assetType as follows:  ASSET ? balance : -balance. |  |
| DOM | This column maps the entity/attribute to YDOM. This column will be blank for synthetic attributes. | Entity: SDM entity Service is mapped to SumInfo YDOM entity.  Attribute: clearedBalance attribute of Account entity is mapped to YDOM currentBalance attribute. |  |
| Type | Applicable only to attributes and Defines their type.  Possible Types:  STRING  DATE  ARRAY<TYPE>  ENUM<TYPE>  NUMBER  TODO PDM: Complete list of available NUMBER types.  For ARRAY and ENUM, the type referenced is either an entity type or an enumeration type defined in SDM. | Transaction attribute date will have DATE type.  Transaction attribute description will have STRING type. | Some of the type definitions can be used for validation purposes too. Example STRING(40). |
| Description | Description of the entity/attribute. |  |  |
| Values | Only applicable to enumeration attributes. Defines set of possible values for enumeration attributes. | CREDIT, DEBIT for transactionType attribute Transaction entity. | This will not be present in the SDM csv files but will be filled when generating external documentation. |
| Default Order | For entities: specifies the attribute that should be used to order the response by default.  For attributes: specifies the default sort order. Possible values:  ASC  DESC | Transaction will have “date” in this column signifying that date should be used to order the response by default. This is applicable even if the date attribute is not requested by the caller.  Transaction attribute “date” will have DESC in this column to specify that by default transactions should be ordered LATEST first. |  |
| Default Filter | For entities: specifies the attribute that should be used to filter the response by default.  For attributes: specifies the value of filter that should be used by default for the attribute.  TODO PDM: Provide list of default filters. Most probably:  [LAST|NEXT] N [DAY|MONTH|YEAR] | By default, we only want to return last 30 days transactions. This can be achieved by specifying date for Transaction entity and “LAST 30 DAY” for date attribute in this column. | TODO PDM: Complete  available filters |
| Create | For entities: specifies if an entity can be created using the API.  “TRUE” indicates the entity can be created. Any other value in the column indicates that some attributes need to meet conditions specified in the column when the entity is created. No value indicates the entity cannot be created manually.  For attributes: not applicable | Goal can be created. Hence, Goal entity will have “TRUE” in this column.  Account can be created only if it is manual. Hence, “manual” attribute should be true when account is created. Hence, this column has value “manual” or “manual=true”. |  |
| Update | For entities: specifies if some or all attributes of an entity can be edited. “TRUE” indicates atleast some attributes are editable. Any other value in the column indicates under what conditions attributes of the entity can be edited. No value indicates the entity is not editable.  For attributes: specifies if an attribute is editable. Can be true only if the corresponding entity is also editable.  “TRUE” indicates the attribute is editable. Any other value in the column indicates under what conditions the attribute can be edited. No value indicates the attribute is not editable. | A transaction can be edited if it is manual. Hence, the column specifies “manual” or “manual=true”  For a transaction, category can be edited, hence “TRUE”.  However, date can be edited only for a manual transaction, hence “manual” or “manual=true” |  |
| Delete | For entities: specifies if an entity can be deleted using the API.  “TRUE” indicates the entity can be deleted. Any other value in the column indicates under what conditions the entity can be deleted. No value indicates the entity cannot be deleted.  For attributes: not applicable | Transaction can be deleted only if it is a manual transaction. Hence, “manual” or “manual=true” |  |
| Validation | Applicable only for editable attributes.  Specifies the rules and validations applied to the input value when the object is added/updated.  Implicit rules like data type and length are not specified as that information is available in the “type” column. | For a goal, goal completion date cannot be in the past. Hence specify “>current date” |  |

# Enumeration Definition Format

Following table describes the format for defining enumerations in SDM.

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Description** | **Examples** | **Notes** |
| Enumeration | Name of the enumeration type | AccountStatus, AccountType, TransactionType |  |
| Value | Enumerated possible values or <DOM> if the enumerated possible values are same as those defined in mapped DOM entity | AccountStatus can be defined with following possible values  ACTIVE  DISABLED  SUSPENDED | Similar to entities, this gives PDM the capability to define the external name for existing and new enumerations. Some of the existing enumerations can be re-named or hidden in SDM |
| DOM | Mapping to YDOM enumeration for entities and to enumerated values for values. | AccountStatus SDM entity can be mapped to AccountStatus DOM entity. |  |
| Type | For later use |  |  |
| Description | Description of each enumerated value |  |  |

# Understanding columns in the SDM

Every SDM spreadsheet contains the following columns. Most are populated by product owners unless specified against the column.

1. Entity: Entity or object exposed through the data model for DAPI. This is the basic element that can be accessed using the API. An entity is defined by its attributes.
2. Attribute: A property of the entity the attribute belongs to. An attribute is defined by data type.
3. Qualifier/Container: Determines conditions under which an attribute is available. For example, in case of accounts and transactions, some attributes are available to specific containers, hence container is a qualifier.
   1. If an attribute is applicable to all qualifier values, the column value is left empty.
   2. Other qualifiers may be country (if some user information is available only for some countries (eg. SSN))
4. Reference: In case of generated fields, this column contains details on how the field is computed.
   1. If the field is generated from a formula, the formula should be specified here.
   2. If the field value is determined from multiple fields based on availability (example transaction date is either postdate or transaction date, based on what is available), that should be specified here.
5. DOM: This is the mapping for an attribute with Yodlee internal fields. This column is filled by engineers.
6. Type: The data type of the attribute. This is populated by engineers.
7. Description: Plaintext description of the attribute. Should be crisp, meaningful and useful. Should be reviewed by techpubs as this is exposed to external developers.
8. Value: In case an attribute has default values, the default value is documented here.
9. Editable: Indicates whether the API allows an attribute value to be updated using the YSL.
10. Repeats: If an attribute repeats for a single object, this column is set to true. For example, fields in the login form repeat for a site object.
11. Validation: The validation that will be applied by Yodlee for an **Editable** field when a developer tries to update the field.
12. Dependency: In case of **Editable** fields, if editing a field mandates updating a related field, that depency is specified in this column.

# Adding Components <rename “Yodlee Data Types”>

Complex Yodlee data types that are reusable. Some examples are Money and Date types.

These complex types may contain several fields that are not required to be exposed to external developers, the Yodlee SDM can be used to expose only useful fields and to eliminate duplicates.

Components are stored here - //razor/core/newarch/modules/sdm/components

# Adding Enumerations

While defining entities, a product owner should specify which attribute is an enumeration so that engineering can create the enumeration objects will all possible values here -

//razor/core/newarch/modules/sdm/enumerations

# Other considerations for populating the SDM

When adding an attribute, go over the following to populate columns

1. Specifying filters: Is it a value based on which the result can be filtered?

Example – a. accounts and transactions can be filtered by container, so container is a filter.

b. Transactions can be filtered by Category, so Category is a filter.

c. Transactions can be filtered by type (credit/debit), so transaction base type is a filter.

To specify that an attribute is a filter, use <filter> in the Type column for that attribute.

1. Specifying component types: If is a standard data type (String, Boolean, etc), engineering can fill this out. For balances and dates, use Components in the type column

Examples:

<COMPONENT>Money

<COMPONENT>Date

1. Entity/attribute names:
   1. Entity and attribute names should be short, meaningful and should not have spelling errors.
   2. Entity and attribute names should use camelcase <or do we recommend any other case?>
   3. Duplicate attributes, unused attributes, unpopulated attributes should be avoided in the data model.
   4. The primary identifier for the object should be exposed as “id”.
2. Review for correctness: All changes to the SDM must be reviewed by
   1. Technology: Rajan, Semyon
   2. PDM: Product owner for the related feature
   3. YSO
   4. Performance
   5. Techpubs

# Backward Compatibility Guidelines

1. Entities, components and enumerations can be added, but cannot be deleted.
2. Attributes can be added to existing entities, components and enumerations. Existing attribute names, type, validation rules, dependencies cannot be modified. Existing attributes cannot be deleted.