DVPD model profile syntax Reference

Credits and license

Overview

DVPD model profile syntax must be supported by any implementation in the same way. If an implementation leaves out elements for simplicity, it should implement a check and warning message, to prevent false expectations.

The syntax must and can be extended by properties needed for project specific solutions (e.g. data_extraction modules, data encryption frameworks). Documentation for these properties must be provided for every module in a separate document.

A DVPD model profile is expressed with JSON syntax and contains the following attributes(Keys):

Root

core elements

dvpd_version (mandatory)
Used to allow checking of compatibility. Must be set to the first version, that supports the used core elements. Minor version changes are kept backwardscompatible. Major version changes might modify structure, keywords and functionality.
"1.0"

model_profile_name (mandatory)
Identifies the profile. The name is referenced by the DVPD property "model_profile_name" at DVPD level and/or table level.

At least the model profile with the name "_default" must be declared in a project. It will be applied to every DVPD, that omits the declaration of a model_profile.

Example: "postgresql_with_enddating"

table_key_column_type (mandatory)
Database column type to be used for the columns with a key hash. This must be a valid SQL type for the database used.
Example: CHAR(28)

table_key_hash_function (mandatory)
Name of the hash function to use when hashing data vault table keys (hub keys, link keys). Valid names depend on the implementation of the staging phase. Recommended values are common lowercase names of the functions (md5, sha-1, sha-256)
Example: sha-1

table_key_hash_encoding (mandatory)
Name of the method to encode the hash value. This can be "hex" (default) or any other method supported by the implementation of the staging. Recommended values are common lowercase names of methods for encoding binary values: (binary, hex, base64)
Example: base64

hash_concatenation_separator (mandatory)
Character or string of characters to be used as separator when concatinating fields before hashing.
Example: |

hash_timestamp_format_sqlstyle (mandatory)
Format description for converting timestamp values into a string before hashing. The syntax follows the sql syntax.
*Example:YYYY-MM-DD HH24:MI:SS.US"

hash_decimal_separator (optional)
Character to be used as decimal separator, when converting numbers with decimals into a string before hashing. Default is ".".
*Example: ."

hash_null_value_string (mandatory)
String to be used in value concatination for hashing, when a field contains a NULL value.
Example: "" (empty string)

key_for_null_ghost_record (mandatory)
Hash value to be used for the ghost record, that will be addressed when all business keys are null in a delivered source record (will happen in optional foreign key relations). The encoding of the value depends on the table_key_hash_encoding.

Beginning and ending this value with $, declares it to be a code snippet, that will be injected in the generated code. Render modules will take this into account, to add necessary quotes when placing it into their target.

Example: "0000000000000000000000000001"
"$lib.hash_key_for_delivered_null()$"

key_for_missing_ghost_record (mandatory)
Hash value to be used for the ghost record, that can be addressed when a business rule cannot find a relation. The encoding of the value depends on the table_key_hash_encoding.

Beginning and ending this value with $, declares it to be a code snippet, that will be injected in the generated code. Render modules will take this into account, to add necessary quotes when placing it into their target.

Example:"FFFFFFFFFFFFFFFFFFFFFFFFFFFE"
"$lib.hash_key_for_missing()$"

content_for_missing_string (mandatory)
String value to be used in the missing ghost record für varchar columns. Columns must accept data with at least this length.

Example:"#missing#" (9 Characters)

content_for_missing_number (mandatory)
Numeric value to be used in the missing ghost record for number columns. Columns must accept data with at least this length.

Example:NULL

content_for_missing_timestamp (mandatory)
Timestamp value to be used in the missing ghost record for timestamp columns.

Example: 1900-01-01 00:00:00

insert_criteria_default (mandatory)
Declares the criteria, that will be checked before a row is inserted. Valid settings are:

- key = the key (hub key or link key) must not be in the satellite
- data = the value combination of the relevant compare columns or the diff hash must not be in the satellite
- current = the value combination of the relevant compare columns or the diff hash are not equal to a current row in the satellite
- key+data = comparison of data also includes the satellites key
- key+current = comparison of current values also includes the satellites key (this is the main mode of data vault satellites)
- none = data will always be inserted (prevention of duplication by repeated loads must be solved by load orchestration)

The settings "key", "current" and "key+current" should be supported by every implementation, since they are needed to declare table load behavior of the Data Vault method. The settings "key" and "none" might remove a declared diff hash column, or at least will leave out the check for a diff_hash even, when uses_diff_hash is true.

uses_diff_hash_default (mandatory)
Determines the default method to compare data rows. When set to true, all historized satellites with content columns must declare a diff hash column. This can be overruled via table specific settings.

Example: true

diff_hash_column_type (mandatory)
Database column type to be used for the diff hash columns. This must be a valid SQL type for the database used.

Example: CHAR(28)

diff_hash_function (mandatory)
Name of the hash function to use when hashing diff hashes. Valid names depend on the implementation of the staging. Recommended values are common lowercase names of the functions(md5, sha-1, sha-256)
Example: sha-1

diff_hash_encoding (mandatory)
Name of the method to encode the diff hash value. This can be "hex" (default) or any other method supported by the implementation. Recommended values are common lowercase names of methods for encoding binary values: (binary, hex, base64)
Example: base64

is_enddated_default (mandatory)
Determines if an enddate column will be added to the satellite and gets updated by the loading processing. This can be overwritten via table specific properties.
*true"

far_future_timestamp (mandatory, when enddating is used)
Timestamp to be used in the enddating column in new (not yet entdated) records.

Beginning and ending this value with $, declares it to be a code snippet, that will be injected in the generated code. Render modules will take this into account, to add necessary quotes when placing it into their target.

Example: 2299-12-30 00:00:00
$lib.far_future_date()$

load_enddate_column_name (mandatory, when enddating is used)
Name of the column, that keeps the insert timestamp of the replacing row.
META_VALID_BEFORE

load_enddate_column_type (mandatory, when enddating is used)
SQL datatype of the column, that keeps the insert timestamp of the replacing row.

Example: TIMESTAMP

load_date_column_name (mandatory)
Name of the column, that keeps the load timestamp of the current row.
*Example: META_INSERTED_AT"

load_date_column_type (mandatory)
SQL datatype of the column, that keeps the load timestamp of the current row.
*Example: TIMESTAMP"

has_deletion_flag_default (mandatory)
Determines if a deletion flag column will be added to satellites by default.
Example:true

deletion_flag_column_name (mandatory)
Name of the column, which marks the current row to indicate, that the data for the business key is technically deleted/not existing in the source any more.
Example: META_IS_DELETED

deletion_flag_column_type (mandatory)
SQL datatype of the column, that flags the current row to indicate, that the data for the business key is technically deleted/not existing in the source any more.
Example: BOOLEAN

record_source_column_name (mandatory)
Name of the column, to store the identification of the data source, that provided the data.
Example: META_RECORD_SOURCE

record_source_column_type (mandatory)
SQL datatype of the column, to store the identification of the data source, that provided the data.
Example: VARCHAR(255)

load_process_id_column_name (mandatory)
Name of the column, to store the identification of the process, that inserted the row.
*META_JOB_INSTANCE_ID"

load_process_id_column_type (mandatory)
SQL datatype of the column, to store the identification of the process, that inserted the row.
*INT8"

elements for the encryption extention

xenc_encryption_key_column_type (mandatory, when using the encryption extention)
SQL datatype of the column, to store the encryption keys.
CHAR(28)

xenc_encryption_key_index_column_type (mandatory, when using the encryption extention)
SQL datatype of the column, to store the encryption key index.

INT8

xenc_content_hash_column_type (mandatory, when using the encryption extention)
SQL datatype of the column, to store the content diff hash in the encryption key tables.
CHAR(28)

xenc_content_hash_function (mandatory, when using the encryption extention)
Name of the hash function to use when hashing diff for the encryption key tables. Valid names depend on the implementation of the staging phase. Recommended values are common lowercase names of the functions (md5, sha-1, sha-256)
sha-1

xenc_content_hash_encoding (mandatory, when using the encryption extention)
Name of the method to encode the content hash value. This can be "binary" (default) or any other method supported by the database and implementation of the staging phase. Recommended values are common lowercase names of methods for encoding binary values: (binary, hex, base64)
BASE64