

# DVPD model profile syntax Reference

---

## Credits and license

(C) Matthias Wegner, cimt ag

Creative Commons License [CC BY-ND 4.0](#)

## 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" an DVPD level and/or table level.

At least one 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 key columns. 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. Recommended values are common lowercase names of the

Example: sha-1

*Example: base64*

Example: |

\*Example:YYYY-MM-DD HH24:MI:SS.US"

\*Example: ."

Example: "" (empty string)

[illegible]

*Example:* "FFFFFFFFFFFFFFFFFFFFFFFFFFFFE"

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)

Timeatmp value to be used in the missing ghost record for timestamp columns.

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

**uses\_diff\_hash\_default** (mandatory)

Determines the default method to determine changes of data. When set to true, all historized satellites with content columns must declare a diff hash column. This can be overwritten via table specific properties.

*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 "binary" (default) or any other method supported by the database and implementation of the staging. Recommended values are common lowercase names of methods for encoding binary values: (binary, hex, base64)

*Example: base64*

**is\_enddated\_default** (mandatory)

Determines if a 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.

*Example: 2299-12-30 00:00:00*

**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 insert timestamp of the current row.

\*Example: META\_INSERTED\_AT"

**load\_date\_column\_type** (mandatory)

SQL datatype of the column, that keeps the insert 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 extension

**xenc\_encryption\_key\_column\_type** (mandatory, when using the encryption extension)

SQL datatype of the column, to store the encryption keys.

*CHAR(28)*

**xenc\_encryption\_key\_index\_column\_type** (mandatory, when using the encryption extension)

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. 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. Recommended values are common lowercase names of methods for encoding binary values: (binary, hex, base64)

*BASE64*