Vital Events Birth, Stillbirth and Death notifications - Tracker Installation Guide { #crvs-ve-trk-installation }

This document includes an installation guide for the updated Vital Events tracker package.

System default language: English

Available translations: French, Spanish, Portuguese

Overview

The package metadata json files contain a "package" component that provides technical details on package version and content. The files available in the current version of the package are listed below.

DHIS2.35

=== "Complete Package"

```
```json
"package": {
    "DHIS2Build": "35d663a",
    "DHIS2Version": "2.35.11",
    "code": "VE0000",
    "description": "Vital Events",
    "lastUpdated": "20220120T113753",
    "locale": "en",
    "name": "CRVS_VE_TKR_1.1.0_DHIS2.35.11-en",
    "type": "TKR",
    "version": "1.1.0"
}
```
```

DHIS2.36

=== "Complete Package"

```
```json
"package": {
    "DHIS2Build": "5d136cb",
    "DHIS2Version": "2.36.6",
    "code": "VE0000",
    "description": "Vital Events",
    "lastUpdated": "20220120T140039",
    "locale": "en",
    "name": "CRVS_VE_TKR_1.1.0_DHIS2.36.6-en",
    "type": "TKR",
```
```

```
    "version": "1.1.0"
}
```

Installation

Installation of the module consists of several steps:

1. Preparing the metadata file with DHIS2 metadata.
2. Importing the metadata file into DHIS2.
3. Configuring the imported metadata.
4. Adapting the program after import

It is recommended to first read through each section of the installation guide before starting the installation and configuration process in DHIS2. Identify applicable sections depending on the type of your import:

1. import into a blank DHIS2 instance
2. import into a DHIS2 instance with existing metadata.

The steps outlined in this document should be tested in a test/staging DHIS2 instance and only then applied to a production environment.

Requirements

In order to install the module, an administrator user account on DHIS2 is required.

Great care should be taken to ensure that the server itself and the DHIS2 application are well secured, access rights to collected data should be defined. Details on securing a DHIS2 system is outside the scope of this document, and we refer to the DHIS2 documentation.

Metadata files

While not always necessary, it can often be advantageous to make certain modifications to the metadata file before importing it into DHIS2.

The Vital Events tracker package includes three metadata files. The contents and purpososse of each individual file are described below:

Package identifier	Contents	Purpose
CRVS_VE_TKR_1.1.0_DHIS2.35.11-en	Updated tracker package	New implementation

Preparing the metadata file

Default data dimension

In early versions of DHIS2, the UIDs of the default data dimensions were auto-generated. Thus, while all DHIS2 instances have a default category option, data element category, category combination and category option combination, the UIDs of these defaults can be different. Later versions of DHIS2 have hardcoded UIDs for the default dimension, and these UIDs are used in the configuration packages.

To avoid conflicts when importing the metadata, it is advisable to search and replace the entire .json file for all occurrences of these default objects, replacing UIDs of the .json file with the UIDs from the instance in which the file will be imported. Table 1 shows the UIDs which should be replaced, as well as the API endpoints to identify the existing UIDs

Object	UID	API endpoint
Category	GLevLNI9wkl	../api/categories.json?filter=name:eq:default
Category option	xYerKDKCefk	../api/categoryOptions.json?filter=name:eq:default
Category combination	bjDvmb4bfuf	../api/categoryCombos.json?filter=name:eq:default
Category option combination	HllvX50cXC0	../api/categoryOptionCombos.json?filter=name:eq:default

Identify the UIDs of the default dimesions in your instance using the listed API requests and replace the UIDs in the json file with the UIDs from the instance.

> **NOTE**
>
> Note that this search and replace operation must be done with a plain text editor, not a word processor like Microsoft Word.

Indicator types

Indicator type is another type of object that can create import conflict because certain names are used in different DHIS2 databases (.e.g "Percentage"). Since Indicator types are defined by their factor (including 1 for "numerator only" indicators), they are unambiguous and can be replaced through a search and replace of the UIDs. This method helps avoid potential import conflicts, and prevents the implementer from creating duplicate indicator types. The table below contains the UIDs which could be replaced, as well as the API endpoints to identify the existing UIDs:

Object	UID	API endpoint
Numerator only (number)	kHy61PbChXr	../api/indicatorTypes.json?filter=number:eq:true&filter=factor:eq:1
Per 1000	zpa0vUC7IWd	../api/indicatorTypes.json?filter=number:eq:false&filter=factor:eq:1000
Percentage	hmSnCXmLYwt	../api/indicatorTypes.json?filter=number:eq:false&filter=factor:eq:100

Tracked Entity Type

Like indicator types, you may have already existing tracked entity types in your DHIS2 database. The references to the tracked entity type should be changed to reflect what is in your system so you do not

create duplicates. The table below contains the UIDs which could be replaced, as well as the API endpoints to identify the existing UIDs:

Object	UID	API endpoint
Person	MCPQUTHX1Ze	../api/trackedEntityTypes.json?filter=name:eq:Person

Visualizations using Root Organisation Unit UID

Visualizations, event reports, report tables and maps that are assigned to a specific organisation unit level or organisation unit group, have a reference to the root (level 1) organisation unit. Such objects, if present in the metadata file, contain a placeholder <OU_ROOT_UID>. Use the search function in the .json file editor to possibly identify this placeholder and replace it with the UID of the level 1 organisation unit in the target instance.

Option codes

According to the DHIS2 naming conventions, the metadata codes use capital letters, underscores and no spaces. Some exceptions that may occur are specified in the corresponding package documentation. All codes included in the metadata objects in the current version of the package were adjusted to match the naming conventions. It may occur that the codes used in the earlier versions of the package used lower case characters. If data values in the existing implementations contain lower case codes, it is important to update those values directly in the database.

The table below contains all option sets where codes were changed to upper case in the metadata package. Before importing metadata into the instance, check whether the option sets in the existing system match those in the package .json and use the same upper case option codes.

Option set name	Option set UID
GEN - Birth attendant type	gHkSQ7ti6zn
GEN - Birth type	jumQ0TEDlf4
GEN - Manner of death	A7mNd2r3ZJe
GEN - Mode of delivery	whFhwY80xAQ
GEN - Place of birth	BkY9x47OEff
GEN - Sex (with unknown)	rlYDq7U043q
Marital status	rkRT5bxwyAt
Relationship (Mother/Father/Spouse/Other)	ocdVHauxjzI
Stillbirth type	tPXEZ46FACM
VE - Place of death occurrence	tPXEZ46FACM
VE - Registration Reason	I90dDKWASnH

The table below contains metadata elements that use an affected option set:

Metadata object	Name	UID
Data element	GEN - Place of birth	ABhkInP0wGY
Data element	GEN - Birth type	LtlzGAPWWo8
Data element	GEN - Attendant at birth	lQtJB35vsDj
Data element	GEN - Mode of delivery	fF7wxNym0Un
Data element	VE - Place of death	XkvdOAv6d3V
Data element	GEN - Manner of Death	MAqI45DkhPd
Data element	VE - Stillbirth classification	vjNZ3tj3ins
Tracked Entity Attribute	Vital Events Sex M/F/U	fSn3gGMwRLi
Tracked Entity Attribute	Vital Events Marital Status	EhEPmB7n31b
Tracked Entity Attribute	Vital Events Relationship 1	Nv4K5ob82z3
Tracked Entity Attribute	Vital Events Relationship 2	egRR3lyqD0F

> **Important**
>
> During the import, the existing option codes will be overwritten with the updated upper case codes. In order to update the data values for existing data in the database, it is necessary to update the values stored in the database using database commands. Make sure to map existing old option codes and new option codes before replacing the values. Use staging instance first, before making adjustments on the production server.

For data element values, use:

```SQL
UPDATE programstageinstance
SET eventdatavalues = jsonb_set(eventdatavalues, '{"<affected data element
uid>","value"}', '"<new value>"')
WHERE eventdatavalues @> '{"<affected data element uid>":{"value": "<old
value>"}}'::jsonb
AND programstageid=<database_programsatgeid>;
```

For tracked entity attribute values, use:

```SQL
UPDATE trackedentityattributevalue
SET value = <new value>
WHERE trackedentityattributeid=<affected trackedentityattribute
database_id> AND value=<old value>;
```

> **Example**
>
> To replace the option code 'yes' with 'YES' for existing data values (data element COVAC - Previously infected with COVID `LOU9t0aR0z7`) in the programstage with the id=1510410385 (example id), the command will be configured as follows:

```SQL
UPDATE programstageinstance
SET eventdatavalues = jsonb_set(eventdatavalues,
'{"LOU9t0aR0z7","value"}', '"YES"')
WHERE eventdatavalues @> '{"LOU9t0aR0z7":{"value": "yes"}}'::jsonb
AND programstageid=1510410385;
```

Option codes are also used in program rule expressions, program indicators, etc. If you are updating code options in your system, make sure you update the codes in all affected metadata objects.

Sort order for options

Check whether the sort order `sortOrder` of options in your system matches the sort order of options included in the metadata package. This only applies when the json file and the target instance contain options and option sets with the same UID.

After import, make sure that the sort order for options within an option set starts at 1. There should be no gaps (eg. 1,2,3,5,6) in the sort order values.

Sort order can be adjusted in the Maintenance app.

1. Go to the applicable Option Set
2. Open the "Options" section
3. Use "SORT BY NAME", "SORT BY CODE/VALUE" or "SORT MANUALLY" alternatives.

Importing metadata

Use Import/Export DHIS2 app to import metadata packages. It is advisable to use the "dry run" feature to identify issues before attempting to do an actual import of the metadata. If "dry run" reports any issues or conflicts, see the import conflicts section below. If the "dry run"/"validate" import works without error, attempt to import the metadata. If the import succeeds without any errors, you can proceed to configuring the module. In some cases, import conflicts or issues are not shown during the "dry run", but appear when the actual import is attempted. In this case, the import summary will list any errors that need to be resolved.

Handling import conflicts

> **NOTE**
>
> If you are importing the package into a new DHIS2 instance, you will not experience import conflicts, as there is no metadata in the target database. After import the metadata, proceed to the "Configuration" section.

There are a number of different conflicts that may occur, though the most common is that there are metadata objects in the configuration package with a name, shortname and/or code that already exist in the target database. There are a couple of alternative solutions to these problems, with different advantages and disadvantages. Which one is more appropriate will depend, for example, on the type of object for which a conflict occurs.

Alternative 1

Rename the existing object in your DHIS2 database for which there is a conflict. The advantage of this approach is that there is no need to modify the .json file, as changes are instead done through the user interface of DHIS2. This is likely to be less error prone. It also means that the configuration package is left as is, which can be an advantage for example when updates to the package are released. The original package objects are also often referenced in training materials and documentation.

Alternative 2

Rename the object for which there is a conflict in the .json file. The advantage of this approach is that the existing DHIS2 metadata is left as-is. This can be a factor when there is training material or documentation such as SOPs of data dictionaries linked to the object in question, and it does not involve any risk of confusing users by modifying the metadata they are familiar with.

Note that for both alternative 1 and 2, the modification can be as simple as adding a small pre/post-fix to the name, to minimise the risk of confusion.

Alternative 3

A third and more complicated approach is to modify the .json file to re-use existing metadata. For example, in cases where an option set already exists for a certain concept (e.g. "sex"), that option set could be removed from the .json file and all references to its UID replaced with the corresponding option set already in the database. The big advantage of this (which is not limited to the cases where there is a direct import conflict) is to avoid creating duplicate metadata in the database. There are some key considerations to make when performing this type of modification:

- it requires expert knowledge of the detailed metadata structure of DHIS2
- the approach does not work for all types of objects. In particular, certain types of objects have dependencies which are complicated to solve in this way, for example related to disaggregations.
- future updates to the configuration package will be complicated.

Configuration

Once all metadata has been successfully imported, there are a few steps that need to be taken before the module is functional.

Sharing

First, you will have to use the *Sharing* functionality of DHIS2 to configure which users (user groups) should see the metadata and data associated with the program as well as who can register/enter data into the program. By default, sharing has been configured for the following:

- Tracked entity type
- Program
- Program stages
- Dashboards
- Visualizations, maps, event reports and report tables
- Data sets
- Category options

Please refer to the DHIS2 documentation for more information on sharing.

Three core user groups are included in the package:

- VE - Access
- VE - Data capture
- VE - Admin

By default, the following permissions are assigned to these user groups:

Object	**User Group**		
	VE - Access	*VE - Data capture*	*VE - Admin*
Tracked entity type	Metadata : can view Data: can view	Metadata : can view Data: can capture and view	Metadata : can edit and view Data: can view
Program	Metadata : can view Data: can view	Metadata : can view Data: can capture and view	Metadata : can edit and view Data: can view
Program Stages	Metadata : can view Data: can view	Metadata : can view Data: can capture and view	Metadata : can edit and view Data: can view
Dashboards	Metadata : can view	No Access	Metadata : can edit and view
Data Sets	Metadata : can view Data: can view	No Access	Metadata : can edit and view Data: No access

The users are assigned to the appropriate user group based on their role within the system. Sharing for other objects in the package may be adjusted depending on the set up. Refer to the DHIS2 Documentation on sharing for more information.

User roles

Users will need user roles in order to engage with the various applications within DHIS2. The following minimum roles are recommended:

1. Tracker data analysis : Can see event analytics and access dashboards, event reports, event visualizer, data visualizer, pivot tables, reports and maps.
2. Tracker data capture : Can add data values, update tracked entities, search tracked entities across org units and access tracker capture

Refer to the DHIS2 Documentation for more information on configuring user roles.

Organisation units

The program and the data sets must be assigned to organisation units within existing hierarchy in order to be accessible via tracker capture/capture apps.

Duplicated metadata

> **NOTE**
>
> This section only applies if you are importing into a DHIS2 database in which there is already meta-data present. If you are working with a new DHIS2 instance, please skip this section and go to Adapting the tracker program. If you are using any third party applications that rely on the current metadata, please take into account that this update could break them.

Even when metadata has been successfully imported without any import conflicts, there can be duplicates in the metadata - data elements, tracked entity attributes or option sets that already exist. As was noted in the section above on resolving conflict, an important issue to keep in mind is that decisions on making changes to the metadata in DHIS2 also needs to take into account other documents and resources that are in different ways associated with both the existing metadata, and the metadata that has been imported through the configuration package. Resolving duplicates is thus not only a matter of "cleaning up the database", but also making sure that this is done without, for example, breaking potential integrating with other systems, the possibility to use training material, breaking SOPs etc. This will very much be context-dependent.

One important thing to keep in mind is that DHIS2 has tools that can hide some of the complexities of potential duplications in metadata. For example, where duplicate option sets exist, they can be hidden for groups of users through sharing.

Adapting the program

Once the program has been imported, you might want to make certain modifications to the program. Examples of local adaptations that *could* be made include:

- Adding additional variables to the form.
- Adapting data element/option names according to national conventions.
- Adding translations to variables and/or the data entry form.
- Modifying program indicators based on local case definitions

However, it is strongly recommended to take great caution if you decide to change or remove any of the included form/metadata. There is a danger that modifications could break functionality, for example program rules and program indicators.

Removing metadata

In order to keep your instance clean and avoid errors, it is recommended that you remove the unnecessary metadata from your instance.

In order to remove the old dashboard from your system, you need to:

> **NOTE**
>
> It is possible to delete the dashboard, its dashboard items and all relevant visualizations, maps and reports directly from the database using SQL commands.

Upgrading from 1.0.0 to 1.1.0

Metadata delete

Importing version 1.1.0 of the Vital Events package in an instance where 1.0.0 has been already installed is enough to create and update the metadata but the following metadata elements should be deleted:

Type	UID	Name
TEA	u0dqjGQ3Z80	VE - Sex is unknown
PR	napRgSZqVp4	Hide Sex if "Sex is unknown" is checked
PR	dp6ev2TTLUH	Hide Sex is unknown if reason for registration is not 2 (stillbirth)

Migrating legacy data to the new metadata

Update values entered for all TEA Sex, i.e. replace UID of TEA Sex with UID of TEA Sex M/F/U

```SQL
UPDATE trackedentityattributevalue
SET trackedentityattributeid = (SELECT trackedentityattributeid FROM
trackedentityattribute where UID = 'fSn3gGMwRLi')
WHERE trackedentityattributeid = (SELECT trackedentityattributeid FROM
trackedentityattribute where UID = 'oindugucx72');
```

For all values corresponding to TEA Sex is Unknown = true, replace UID of TEA with UID of TEA Sex M/F/U and replace true with UNKNOWN

```SQL
UPDATE trackedentityattributevalue
SET value = 'UNKNOWN',
trackedentityattributeid = (SELECT trackedentityattributeid FROM
trackedentityattribute where UID = 'fSn3gGMwRLi')
WHERE trackedentityattributeid = (SELECT trackedentityattributeid FROM
trackedentityattribute where UID = 'u0dqjGQ3Z80')
and value = 'true';
```