VPD CS Tracker Installation Annex { #vpd-cs-installation }

Overview

The VPD CS tracker package is self-contained, meaning it can be installed on its own. However, it is worth noticing that this package contains some metadata from the IDSR package. The reason for this is that there is a comparative table which makes it possible to compare the aggregated data and the tracker data. Thus, we recommend installing the aggregate disease surveillance ("IDS") package prior to this one, although it is not a mandatory requirement.

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 being imported

It is recommended to first read through each section before starting the installation and configuration process in DHIS2. Sections that are not applicable have been identified, depending on if you are importing into a new instance of DHIS2 or a DHIS2 instance with metadata already present. The procedure outlined in this document should be tested in a test/staging environment before either being repeated or transferred to a production instance of DHIS2.

Requirements

In order to install the module, an administrator user account on DHIS2 is required. The procedure outlined in this document should be tested in a test/staging environment before being performed on a production instance of DHIS2.

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

Preparing the metadata file

NOTE: If you are installing the package on a new instance of DHIS2, you can skip the "Preparing the metadata file" section and move immediately to the section on "Importing a metadata file into DHIS2."

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

Default data dimension

In early versions of DHIS2, the UID of the default data dimension was 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 of the database 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. 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`

For example, if importing a configuration package into https://play.dhis2.org/demo, the UID of the default category option combination could be identified through https://play.dhis2.org/demo/api/categoryOptionCombos.json?filter=name:eq:default as bRowv6yZOF2.

You could then search and replace all occurrences of HllvX50cXC0 with bRowv6yZOF2 in the .json file, as that is the ID of default in the system you are importing into. 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 simply by their factor and whether or not they are simple numbers without a denominator, they are unambiguous and can be replaced through a search and replace of the UIDs. This avoids potential import conflicts, and avoids creating duplicate indicator types. Table 2 shows the UIDs which could be replaced, as well as the API endpoints to identify the existing UIDs

Object	UID	API endpoint
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. Table 3 shows 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.

Generic metadata

The package contains some metadata that is typically common or shared across many health programs. These metadata have a prefix 'GEN' to signify that they are generic and re-usable in other parts of the DHIS2 configuration. It is likely that you have similar elements already in your database. These generic elements are used in some indicators and they are included in the aggregate surveillance/IDSR metadata file.

Generic Data Elements used in the VPD CS package

Object	UID	API endpoint
GEN - Population	DkmMEcubiPv	`../api/dataElements.json?filter=name:like:GEN - Population`
GEN - Population < 15years	cPLAnOTIdta	`../api/dataElements.json?filter=name:like:GEN - Population`
GEN - Population live births	iKxybad85rv	`../api/dataElements.json?filter=name:like:GEN - Population`
GEN - Population weekly	iLEkjJcYTJd	`../api/dataElements.json?filter=name:like:GEN - Population`

Category options, categories, category Combos for ages

Object	UID	API endpoint
0-4 years	UPvKbcqTEY3	`../api/categoryOptions.json?filter=name:like:years`
5+ years	afpMrUgPzl3	`../api/categoryOptions.json?filter=name:like:years`
Age (surveillance)	wAZr8lv7S64	`../api/categories.json?filter=name:like:Age`
Age (surveillance)	eg8enaFY861	`../api/categoryCombos.json?filter=name:like:Age`

Predictors using Organisation Unit Level district

There are many predictors in the package which are using a placeholder for district Organisation Unit Level (OUL). The API endpoint for them would be: `../api/predictors?filter=organisationUnitLevels:!null`

The placeholder uses the label <OU_LEVEL_DISTRICT_UID>. Before attempting to import the package you need to replace this label with the UID of the equivalent OUL in your system.

Importing metadata

The .json metadata file is imported through the Import/Export app of DHIS2. 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 configure 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.

Generic metadata

For generic metadata overview (identical metadata objects used by several packages), please refer to the installation annex of the individual package.

Handling import conflicts

NOTE: If you are importing into a new DHIS2 instance, you will not have to worry about import conflicts, as there is nothing in the database you are importing to to conflict with. Follow the instructions to import the metadata then please proceed to the “Additional 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 exists 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 training material and documentation based on the configuration package will be used.

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.

Additional configuration

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

Sharing

There are three user groups that come with the package:

- VPD CS access
- VPD CS admin
- VPD CS data capture

By default the following is assigned to these user groups

Object	User Groups		
	VPD CS access	VPD CS admin	VPD CS data capture
Tracked entity type	Metadata : can view Data: can view	Metadata : can edit and view Data: can view	Metadata : can view Data: can capture and view
Program	Metadata : can view Data: can view	Metadata : can edit and view Data: can view	Metadata : can view Data: can capture and view
Program Stages	Metadata : can view Data: can view	Metadata : can edit and view Data: can view	Metadata : can view Data: can capture and view
Dashboards	Metadata : can view Data: can view	Metadata : can edit and view Data: can view	Metadata : can view Data: can view

You will want to assign your users to the appropriate user group based on their role within the system. You may want to enable sharing for other objects in the package depending on your set up. Refer to the DHIS2 Documentation for more information on configuring sharing.

Important note: there is a bug affecting versions 2.33 and 2.34 which overrides sharing properties of an object after importing the same object even though MERGE is selected as import parameter. This is especially critical for the object Tracked Entity Type - Person (UID: MCPQUTHX1Ze) which is typically shared with user groups of different packages.

If your TET Person already matches the one in the package and you want to avoid overriding the current sharing properties for this object in the database, you can follow these steps:

- Remove this TET from the package file
- Import the package
- Update sharing properties for this TET using the table above.

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

You must assign the program to organisation units within your own hierarchy in order to be able to see the program in tracker capture.

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."

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 tracker program

Once the programme has been imported, you might want to make certain modifications to the programme. 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.