Immunization (EPI) Triangulation Installation Guide { #imm-tri-installation }

This is the installation guide for the Immunization (EPI) Triangulation package.

System default language: English

Overview

The metadata reference and metadata json files provide technical details on package version and content. The metadata package consists of one module, which is a **dashboard** package.

This package has been designed to provide indicators and analytics for triangulation of data from three different packages:

- Immunization (EPI) aggregate package for HMIS
- Integrated Disease Surveillance (IDS) aggregate package
- Case-based tracker for vaccine-preventable diseases (VPDs)

While this dashboard package do not require that these packages are already installed and used, it is important that it is installed in a DHIS2 instance where both routine immunization data and surveillance data are available. In countries where these programmes are collected and managed in separate DHIS2 instance, it is thus necessary to first bring the data together in one place. This is further explained in the design guide.

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 metadata 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.

Preparing the metadata file

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

> **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 a 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)	`CqNPn5KzksS`	`../api/indicatorTypes.json?filter=number:eq:true&filter=factor:eq:1`

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 package.

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

Three core user groups are included in the packages:

- Immunization access (view metadata/view data)

- Immunization admin (view and edit metadata/no access to data)
- Immunization data capture - (view metadata/capture and view data). This is not used, since this is a dashboard-only package.

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.

Indicator mapping

When implementing the *dashboard package* only, the indicator numerators and denominators have to be configured using the metadata objects in the existing instance. Configuration information is available in the documentation and the description of numerators and denominators in the metadata file. The indicators have a "[CONFIG]" prefix to their name for easy identification, which should removed when the indicator has been configured.

Removing metadata

In order to keep your instance clean and avoid errors, it is recommended that you remove the unnecessary metadata from your instance. Removing unnecessary metadat requires advanced knowledge of DHIS2 and various dependenies.