TB Case Surveillance Tracker Installation Guide { #tb-cs-installation }

Package Version 2.0.0

System default language: English

Available translations: French, Spanish, Portuguese

Installation

Installation of the module consists of several steps:

1. Preparing the metadata file.
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 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 (No other versions of TB Case Surveillance tracker imported previously).
3. Update existing/older version of the TB Case Surveillance tracker.

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, a DHIS2 administrator user account 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 are outside the scope of this document, and we refer to the DHIS2 documentation.

Metadata files

The metadata reference and metadata json files provide technical details on package version and content.

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

Preparing the metadata file

It is recommended to import the DHIS2 Common HIS metadata library into the target instance before using and adapting any DHIS2 metadata packages. Common HIS Metadata package is available for download in the supported versions of DHIS2 at Metadata Package Downloads

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

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`

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 package match the naming conventions. It may occur that the codes of existing metadata objects used in the target database use lower case characters. In this case, it is important to update those values directly in the database.

> **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"}', '"<
WHERE eventdatavalues @> '{"<affected data element uid>":{"value": "<old value>"}}'::jsonb
AND programstageid=<database_programsatgeid>;
```

> **NOTE**
>
> When updating the UID of a metadata element in the existing DHIS2 instance, you will need to run an SQL command in the database and additionally replace all occurrences and references of its UID in other metadata objects: predictor, indicator, validation rule expressions, etc.

Sort order of 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 of 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.

Make sure that no options within an option set have the same sort order. This can be checked using the following api endpoint:

`../api/options.json?paging=false&fields=id,name,sortOrder&filter=optionSet.id:in:[<optionSet UID>]`

In order to fix sort order in option sets containing large numbers of options, please refer to this SQL script.

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.

Some visualizations and maps may contain references to organisation unit levels. Maps that consist of several map views may contain vaious Organisation unit level references based on the configuration of the map layer. Adjust the organisation unit level references in the metadata json file to match the organisation unit structure in the target instance before importing the metadata file.

Upgrading metadata package

The process of upgrading an existing package to a newer version in a working DHIS2 instance is a complex operation that has to be taken with precaution. Such process has to be run in a staging instance first, before upgrading the configuration on the production server. As metadata objects may have been removed, added or changed, it is important to ensure that:

- the format of existing data can be mapped and adjusted to the new configuration;
- the discontinued metadata objects are deleted from the instance;
- The existing objects are updated;
- the new objects are created;
- assignment of users to relevant user groups is reviewed.

The diff file will help the implementer identify the necessary changes.

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.

Upgrading earlier versions of TB CS Tracker program

This section provides guidance on upgrading earlier versions of TB CS tracker (eg. versions 1.0.0 or 1.0.1).

For existing implementations, direct upgrade of metadata packages in the instance is not recommended.

Use the following steps before choosing the best upgrade strategy:

1. Import **TB CS tracker 2.0.0** into a new test instance
2. Compare and map existing data with the TB CS tracker 2.0.0 configuration
3. Adapt TB CS tracker 2.0.0 based on local requirements
4. Use existing or custom data transfer tools to transfer data from existing configuration to TB CS tracker 2.0.0
5. Make assessment-based decisions regarding upgrade stratagy (adapting existing configuration or moving data to the new tracker)

For demo and training purposes, the following upgrade process is recommended:

1. Create a backup of the instance with the TB CS tracker
2. The TB CS metadata objects that are listed below will have to be deleted from the target instance before importing TB CS tracker.2.0.0. If you have customised or added any metadata objects from the previous versions of the package, these need to be backed up, reconfigured and imported after upgrade. Delete the following metadata objects from the target instance:
    - dashboards
    - visualizations
    - maps
    - eventReports
    - programIndicatorGroups
    - program indicators
    - indicatorGroups
    - indicators
    - trackedEntityInstanceFilters
    - dataElementGroups
    - programRules
    - programRuleActions
    - programRuleVariables
3. If you have existing data in the demo/training instance, you will be required to export all TB CS tracker enrollment data, delete it in the instance and reimport once the data has been mapped and the TB CS tracker has been updated upgraded. In TB CS tracker 1.0.0, only diagnosed TB cases are enrolled in the program. TB CS tracker 2.0.0 allows both presumtive and diagnosed cases to be enrolled. The enrollment date in 1.0.0 (Date of diagnosis) becomes date of registration in 2.0.0. For existing enrollments, the enrollment date has to be mapped with the Date of diagnosis, which is a data element in the Registration stage. Additional mappings for Notification date and other data elements will be required.
4. Import TB CS tracker 2.0.0 into the instance.
5. Import mapped and missing data.
6. Test data entry, dashboards and general functionality.

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:

- Dashboards
- Visualizations, maps, event reports and report tables
- Data sets
- Category options
- Programs and program stages

These core user groups are included in the package:

- TB admin
- TB access
- TB lab access
- TB data capture
- TB lab data capture

By default the following is assigned to these user groups

Object	**User Groups**				
	TB access	TB lab access	TB admin	TB data capture	TB lab data capture
Tracked entity type	**Metadata:** can view **Data:** can view	**Metadata:** can view **Data:** can view	**Metadata:** can edit and view **Data:** no access	**Metadata:** can view **Data:** can capture and view	**Metadata:** can view **Data:** can capture and view
Program	**Metadata:** can view **Data:** can view	**Metadata:** can view **Data:** can view	**Metadata:** can edit and view **Data:** no access	**Metadata:** can view **Data:** can capture and view	**Metadata:** can view **Data:** can capture and view

Object	User Groups				
Program Stages	**Metadata:** can view **Data:** can view	**Group access is limited to stages: TB Registration, Diagnostic and Laboratory Results** **Metadata:** can view **Data:** can view	**Metadata:** can edit and view **Data:** no access	**Metadata:** can view **Data:** can capture and view	**Group access is limited to stages: TB Registration and Laboratory Results** **Metadata:** can view **Data:** can capture and view
Dashboards	**Metadata:** can view **Data:** can view	**Metadata:** can view **Data:** can view	**Metadata:** can edit and view **Data:** no access	No access	No access

Users need to be assigned to the aplicable user group based on their role within the system. Sharing for other objects in the package should be set up depending on requirements. Refer to the DHIS2 Documentation for more information on configuring sharing.

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

Program must be assigned to applicable organisation units within the organisation unit hierarchy.

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.

Constants

TB Case Surveillance Tracker package includes a set of tests and a list of drugs that can be modified by the implementing country according to national context (e.g. which drugs and tests are used/available in country). The use of constants and corresponding program rules enables a system admin in an implementing country to easily ‘turn on’ or ‘turn off’ types of drugs and tests depending on requirements. Instructions for configuring constants are provided in the description of the constant objects.

Configuring tracker capture interface, widgets and top bar

You must configure tracker capture dashboard after the package has been installed. This configuration includes data entry forms, widgets and top bar.

Data entry forms

- After registering the first (test) case, access the **Settings** menu in the tracker capture form and select **Show/Hide Widgets**
- Use **Timeline Data Entry**
- Make sure that **Enrollment**, **Feedback** and **Profile** widgets are selected. Click **Close**.

Top Bar

- Access the **Settings** menu and select **Top bar settings**
- Select **Activate top bar**
- Select required information fields and assign **Sort order**

Recommended fields	**Sort order**
Attributes	
TB Registration Number	2
Indicators	
Case classification	8
Patient's age (years)	5
HIV Status at diagnosis	6

Recommended fields	Sort order
Resistance at diagnosis	10
Treatment regimen	9
Date of diagnosis	1
Months since diagnosis	3
Resistance classification at diagnosis	7
Patient's age (months)	4

- Click **Save**
- Return to the **Settings** menu. Click **Saved dashboard layout as default**. Lock layout for all users.

Program notifications

TB CS Tracker 2.0.0 includes 4 notification templates that can be edited based on local requirements. Additional configuration for setting up SMS or email notifications is required.

Reporting case-based data into aggregate data sets

The TB case-based surveillance tracker captures data that can be fed into standard, aggregate reporting (i.e. monthly, quarterly, yearly or as determined by the country). Aggregate DHIS2 TB HMIS system design can be accessed at https://dhis2.org/metadata-package-downloads/#tb

This Tracker-to-Aggregate tool can be easily configured for data transfer..

More information is available in the Tracker to aggregate data integration guide.

The **program indicators** in the TB Case Surveillance and Laboratory package are mapped with **data elements** and **category option combinations** in the DHIS2 TB aggregate package (Laboratory, Notifications and outcomes).

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.