IDS - Integrated Disease Surveillance Installation Guide { #ids-agg-installation }

This document includes an installation guide for the IDS aggregate package.

System default language: English

Available translations: French

Overview

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

The metadata package consists of the following modules:

- Integrated Disease Surveillance
- Integrated Disease Surveillance (dashboard)

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.

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`

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 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:

```
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>;
```

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.

The IDS package contains one option set and two options:

```
{
    "optionSets": [
```

```
        {
            "name": "YES/NO (numeric)",
            "id": "TdDqpX1kdd2",
            "code": "YES_NO_NUM",
            "valueType": "INTEGER_ZERO_OR_POSITIVE",
            "options": [
                {
                    "id": "VavIEUmBv8j"
                },
                {
                    "id": "Xu8ieCbS7jH"
                }
            ]
        }
    ],
    "options": [
        {
            "name": "Yes",
            "id": "VavIEUmBv8j",
            "code": "1",
            "sortOrder": 1,
            "optionSet": {
                "id": "TdDqpX1kdd2"
            }
        },
        {
            "name": "No",
            "id": "Xu8ieCbS7jH",
            "code": "0",
            "sortOrder": 2,
            "optionSet": {
                "id": "TdDqpX1kdd2"
            }
        }
    ]
}
```


This Yes/No option set is based on "INTEGER_ZERO_OR_POSITIVE" option values that are assigned to two data elements and evaluated in a predictor listed below:

Type	Name	UID	Details
Data element	IDS One death from severe AWD in a person > 5 years old	`v2FeCwrKNt5`	Used in data sets: `m9yPuQEqxmS`, `ZyZmZTUwctj`
Data element	IDS Two or more AWD aged 2 years and older (linked by time and place) with severe dehydration or dying	`ZOVJGcrCdCH`	Used in data sets: `m9yPuQEqxmS`, `ZyZmZTUwctj`
Predictor	IDS - Cholera/AWD Alert	`nrJQj0Kxp03`	Generator: `if( #{ZOVJGcrCdCH}>= 1`

Population data

The IDS package includes data elements, indicators and other metadata objects that are related to **population** data.

Data element	UID
GEN - Population	`DkmMEcubiPv`
GEN - Population weekly	`iLEkjJcYTJd`
GEN - Population < 15years	`cPLAnOTldta`

If the target instance already has metadata infrastructure, which is used for collecting **Population data**, please refer to the steps listed below:

1. Choose the strategy to align population metadata in the target instance and in the .json file.
    - Alternative 1: Replace the UIDs of the data elements and all their occurences in the json file with the UIDs from the target system
    - Alternative 2: Consider replacing the UIDs of these data elements in the target system with the UIDs from the json file. GEN data elements are part of DHIS2 core metadata library and are used in other metadata packages.
2. Indicators that use the **population data** will be aggregating data at the level/levels where the data is entered.
3. Additional mapping and configuration may be required after the package is imported. Refer to the data set configuration section

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

Predictors { #idsr-predictors }

The package includes the following predictors:

Name	UID	Period type	Missing value strategy	Output data element - name	Output data element - UID	Organisatio unit levels
IDS - Acute Flacid Paralysis outbreak	`vclhob0deoe`	Weekly	Skip if all values are missing	IDS - Acute Flacid Paralysis outbreak	`yCjneRUc565`	District level

Name	UID	Period type	Missing value strategy	Output data element - name	Output data element - UID	Organisatio unit levels
IDS - Cholera/AWD Alert	`nrJQj0Kxp03`	Weekly	Skip if all values are missing	IDS - Cholera/AWD Alert	`k2hYvS8LTFl`	District level
IDS - Cholera outbreak	`b64lroD7kZI`	Weekly	Skip if all values are missing	IDS - Cholera outbreak	`fVYqPV4Yfuv`	District level
IDS - Dengue Fever Alert	`r64RbaAOIri`	Weekly	Skip if all values are missing	IDS - Dengue Fever Alert	`j3t0jhGOHhP`	District level
IDS - Diarrhoea with blood Alert	`Frp6BqdLOsQ`	Weekly	Skip if all values are missing	IDS - Diarrhoea with blood Alert	`ZYvr5OITjBa`	District level
IDS - Diphtheria Alert	`eZad6JpXT0E`	Weekly	Skip if all values are missing	IDS - Diphtheria Alert	`qFGp7kAZ5Me`	District level
IDS - Measles Confirmed Outbreak	`sieOmlObW8L`	Weekly	Skip if all values are missing	IDS - Measles Confirmed Outbreak	`Z1lCDUuuuHg`	District level
IDS - Measles Suspected Outbreak	`C1ocEPMZUWn`	Weekly	Skip if all values are missing	IDS - Measles Suspected Outbreak	`mZ08SCurQVX`	District level

Name	UID	Period type	Missing value strategy	Output data element - name	Output data element - UID	Organisatio unit levels
IDS - Meningites Alert	`qN9hTkwE4Ye`	Weekly	Skip if all values are missing	IDS - Meningites Alert	`KNruIU8QeKP`	District level
IDS - Meningitis outbreak	`HXBvKxWaujs`	Weekly	Skip if all values are missing	IDS - Meningitis outbreak	`I3570OWVcKN`	District level
IDS - Neonatal Tetanus Alert	`iuNfzUx351B`	Weekly	Skip if all values are missing	IDS - Neonatal Tetanus Alert	`RGXZgVrOQyo`	District level
IDS - Non Neonatal tetanus Alert	`SQ5BUEZpsgd`	Weekly	Skip if all values are missing	IDS - Non Neonatal tetanus Alert	`f45l3xKDAPs`	District level
IDS - Pertussis Alert	`BlwExnIXNEA`	Weekly	Skip if all values are missing	IDS - Pertussis Alert	`I5V4NUxtTre`	District level
IDS - Rabies Alert	`mdYA6Hfl2J4`	Weekly	Skip if all values are missing	IDS - Rabies Alert	`f3MTBlkGjZw`	District level
IDS - Viral Haemorragic Fever Alert	`XhrFVuATU9L`	Weekly	Skip if all values are missing	IDS - Viral Haemorragic Fever Alert	`BSyp9DU4HWn`	District level

Name	UID	Period type	Missing value strategy	Output data element - name	Output data element - UID	Organisatio unit levels
IDS - Yellow Fever Alert	`hub5RjpxB7b`	Weekly	Skip if all values are missing	IDS - Yellow Fever Alert	`jUaZniVe1Uq`	District level

Predictor metadata includes organisation unit levels used for aggregation of data values. The package metadata file contains placeholders that need to be replaced with the UIDs of the corresponding organisation unit levels in the target database.

The steps to prepare the predictors for import are described below:

1. Identify the organisationUnitLevel UID of the District level at which the data for the predictors will be aggregated. Use the following API endpoint to identify the required UID: `../api/organisationUnitLevels.json?fields=id,name`
2. Find the following organisationUnitLevel placeholders in the json file: `<OU_LEVEL_DISTRICT_UID>`
3. Replace the placeholders with the UID of the identified facility level in the target instance.

Validation rules { #idsr-validation-rules }

All validation rules included in the package are listed in the metadata reference file.

The organisation unit groups for all validation rules are set to the district level. The district level value is located in the `"organisationUnitLevels"` property of each validation rule. It is set to `3` by default. Adjust these levels in the metadata file to match the district level in the target instance before importing the package.

Validation rule notifications { #idsr-validation-notifications }

All validation notifications included in the package are listed in the metadata reference file.

Note that all validation rule notifications for this package are all set to be sent as a "single notification." This means that one notification is sent out for each organisation unit/period combination when a notification is triggered. This can also be configured as a "collective summary." The strategy for sending these notifications is located in the `"sendStrategy"` property of each validation notification. It is set to `SINGLE_NOTICIATION` by default. Adjust these values to `COLLECTIVE_SUMMARY` within the metadata file if you would like to change this strategy before importing the package.

NB: Note that you can also change this in maintenance for each validation rule notification at any time after they are imported as you may want to demonstrate or test both strategies to select one that is appropriate for your own setting.

Importing metadata

Use the 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:

- 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:

- IDS access (view metadata/view data)
- IDS admin (view and edit metadata/no access to data)
- IDS alerts (view metadata/capture and view data)

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. Aggregate data analysis : Can access dashboards, data visualizer, pivot tables, reports and maps.
2. Aggregate data capture : Can acccess the data entry app and add/modify data values,

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

Organisation unit assignment

The data sets must be assigned to organisation units within existing hierarchy in order to be accessible for data entry and data analysis personnel.

Creating jobs in the scheduler { #idsr-scheduling }

You will have to use the scheduler app in order to take advantage of the predictor and validation notification components of the package. You will need at least 3 jobs in the following order:

1. Predictor

2. Analytics Table
3. Monitoring

You will want each job to complete before you run the next one (ie. the predictors should all be created before analytics starts; the monitoring job should only start after analytics is complete). Each DHIS2 implementation will need to review their configuration to determine the time it takes to run each of these jobs and schedule them accordingly. It is recommended that you have these run late at night when there is not much activity within your DHIS2 instance, as these are generally resource heavy operations.

A couple tips for each job type:

Predictor

Predictor jobs consist of a relative start and end date. This means you can run your predictors for the most recent period to generate the latest data that you need. This should be useful if your previous data is not being changed, as the other predicted values will already be generated and stored (and thus this process will not necessarily need to occur once more for those already generated values that are still valid). This is a particularly resource heavy operation, and if your previous data is not routinely changing, generating data for the most recent period that you need data for is the recommended approach.

You are also able to select specific predictors or predictor groups to run during the job. The predictor group for this package is simply called `IDS`. If you select multiple groups it will run the predictors in the order the groups are selected. You can read more about this within the DHIS2 documentation.

This needs to be the first job that starts in your sequence, and should finish before the next job (analytics) starts.

Analytics table

The analytics table job takes all of the raw data that has been entered and applies the necessary aggregation to it based on your configuration. If you are using an integrated system with multiple programs inside your instance, then you may already have an analytics table job scheduled to run at routine intervals. If this is the case, you may need to modify the period in which it runs so it can run after the predictor job has been completed.

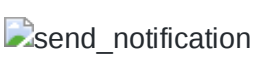
This needs to be second job in your sequence, and should finish before the next job (monitoring) starts.

Monitoring

Similar to the predictor, the monitoring job also consists of a relative start and end date. If your data is not changing during previous periods, you can run the monitoring job only for the period you need to review.

You can specify validation rule groups for the monitoring job. Create one monitoring job that runs weekly, using the validation rule group `IDS - Weekly`

If you want to send out the validation notifications, ensure that the "Send notifications" item is selected.



Sending out notifications

By default, notifications will be sent out to via the DHIS2 messaging app.

In order to send out notifications via SMS or e-mail, you will need an SMS gateway and/or an e-mail configuration set up within your DHIS2 instance. The settings necessary for these will vary between implementations and it is best to consult the documentation links provided here for more information.

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 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 names according to national conventions.
- Adding translations to variables and/or the data entry form.
- Modifying indicators based on local case definitions
- Adding dashboards/dashboard items

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 predictors or notifications.

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 metadata requires advanced knowledge of DHIS2 and various dependencies.