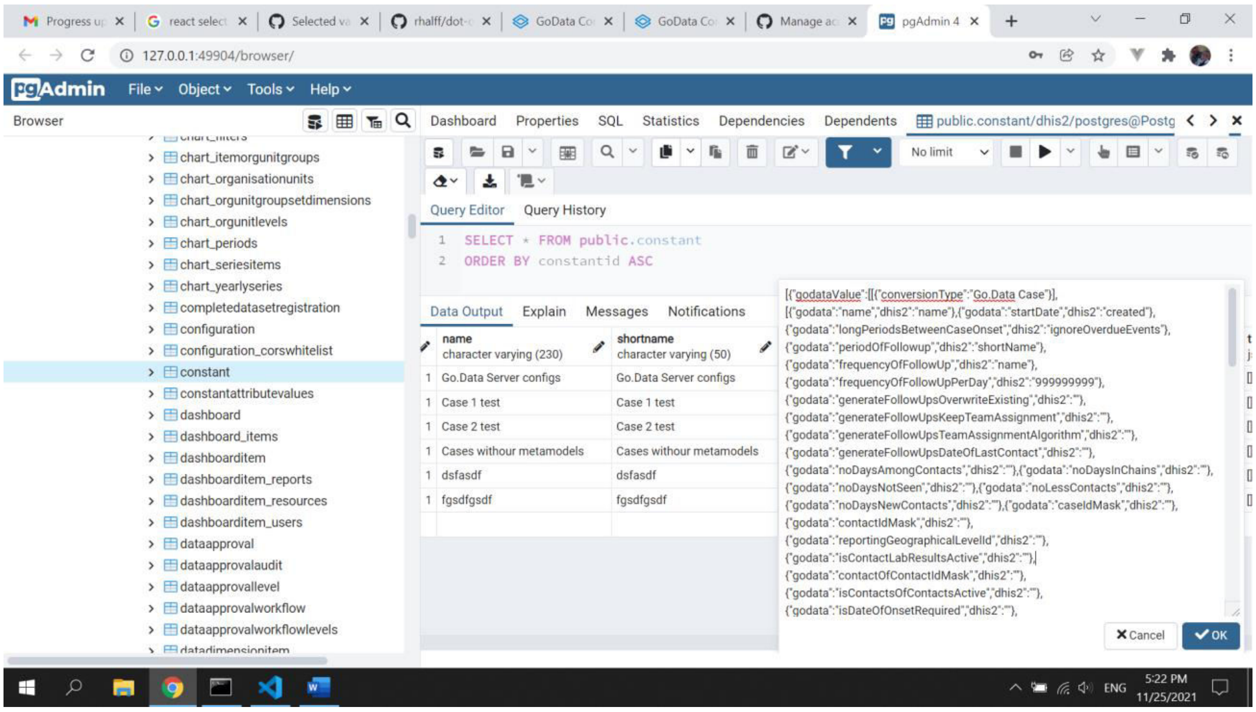
Go.Data <=> DHIS2

interoperability DHIS2 module user manual.

Overview

Development of Go.Data – DHIS2 interoperability to share various metadata and content is based on DHIS2 model framework. For versioning and sharing source code, GihHub repository was created. This repository is marked private at this time and later could be set open for community development on client will. Development environment was set up using MS Visual Studio and linked to GitHub repository. DHIS2 instance was installed in local computer with default WHO COVID-19 metadata. Go.Data instance was installed on local computer as a server. Besides this access to the test server of Go.Data was provided by client.

With this setup development of module started. As DHIS2 modules cannot add new tables to database schema, it was decided to use "constant" table for storing module data. "constant" table has few columns and designed for storing numeric value. These columns are: name, shortName, code, description, translations, attributevalues, useraccess, value. Most of these columns have restrictions and are limited in use. For instance, columns name, shortNme, code are short in size and their value should be unique. Jsonb columns (translations, attributevalues, useraccess) are validated against specific objects in DHIS2 schema and cannot store other objects or documents. The only column useful is description. Description column is of type "text" and can hold large amount of textual data. I used this column for storing all objects used by module in stringified js object notation (JSON) and arrays. This provided a lot of flexibility and at the same time a little overhead for validating objects on and from database read/writes. So "description" column of table "constant" of DHIS2 became "nano mongo DB".



chart_itemorgunitgroups
chart_organisationunits
chart_orgunitgroupsetdimensions
chart_orgunitlevels
chart_periods
chart_seriesitems
chart_yearlyseries
completedatasetregistration
configuration
configuration_corswhitelist
constant
constantattributevalues
dashboard
dashboard_items
dashboarditem
dashboarditem_reports
dashboarditem_resources
dashboarditem_users
dataapproval
dataapprovalaudit
dataapprovallevel
dataapprovalworkflow
dataapprovalworkflowlevels
datadimensionitem

```
SELECT * FROM public.constant
ORDER BY constantid ASC
```

	name character varying (230)	shortname character varying (50)
1	Go.Data Server configs	Go.Data Server configs
1	Case 1 test	Case 1 test
1	Case 2 test	Case 2 test
1	Cases withour metamodels	Cases withour metamodels
1	dsfasdf	dsfasdf
1	fgsdfgsdf	fgsdfgsdf

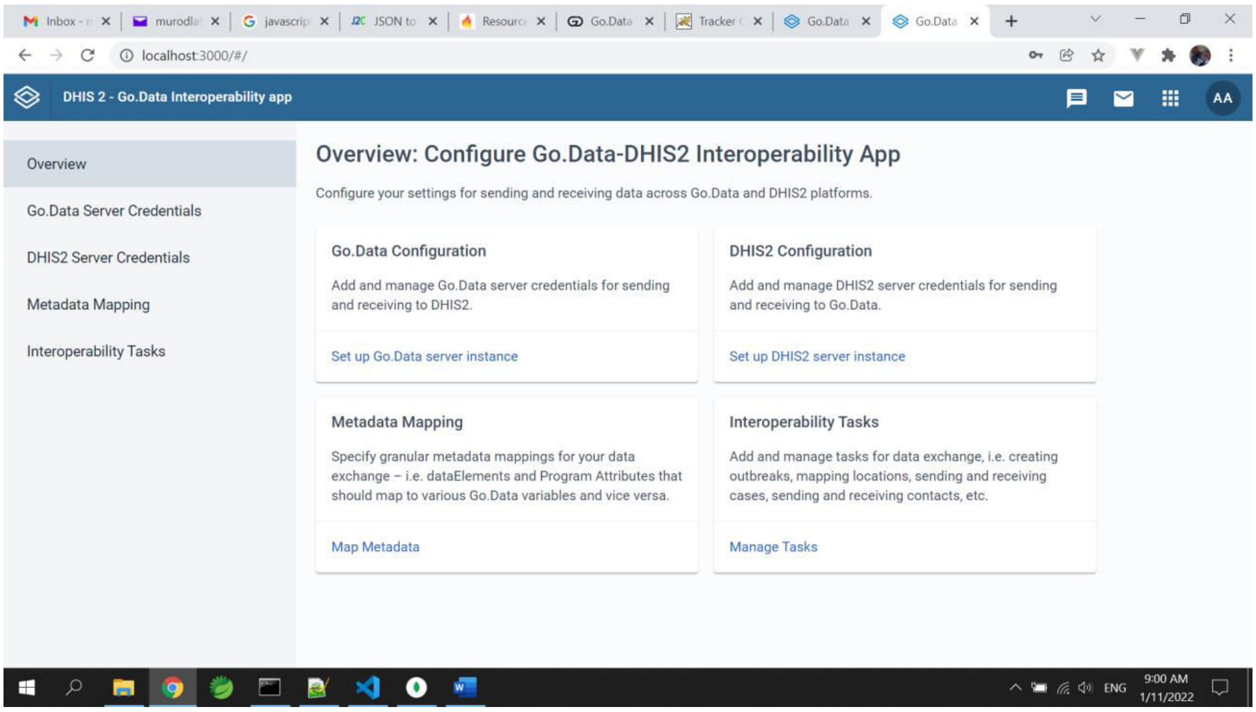
```
[{"godataValue":[[{"conversionType":"Go.Data Case"}],
[{"godata":"name","dhis2":"name"},{"godata":"startDate","dhis2":"created"},
{"godata":"longPeriodsBetweenCaseOnset","dhis2":"ignoreOverdueEvents"},
{"godata":"periodOfFollowup","dhis2":"shortName"},
{"godata":"frequencyOfFollowUp","dhis2":"name"},
{"godata":"frequencyOfFollowUpPerDay","dhis2":"999999999"},
{"godata":"generateFollowUpsOverwriteExisting","dhis2":""},
{"godata":"generateFollowUpsKeepTeamAssignment","dhis2":""},
{"godata":"generateFollowUpsTeamAssignmentAlgorithm","dhis2":""},
{"godata":"generateFollowUpsDateOfLastContact","dhis2":""},
{"godata":"noDaysAmongContacts","dhis2":""},{"godata":"noDaysInChains","dhis2":""},
{"godata":"noDaysNotSeen","dhis2":""},{"godata":"noLessContacts","dhis2":""},
{"godata":"noDaysNewContacts","dhis2":""},{"godata":"caseIdMask","dhis2":""},
{"godata":"contactIdMask","dhis2":""},
{"godata":"reportingGeographicalLevelId","dhis2":""},
{"godata":"isContactLabResultsActive","dhis2":""},
{"godata":"contactOfContactIdMask","dhis2":""},
{"godata":"isContactsOfContactsActive","dhis2":""},
{"godata":"isDateOfOnsetRequired","dhis2":""},
```

Figure 1. "constant" table of DHIS2 showing "description" column

Column value was used to group different object types. For example, value -1000000 is used for configuring Go.Data instance, -1000001 used for mapping of Go.Data metadata with DHIS2 metadata, -1000002 is used for creating tasks. Columns name and shortName are used for naming created entities.

Menu system

User interface of the module consists of left pane with menu options to navigate and main pane for performing and viewing various actions within the module. There are 4 menu options as shown in figure 2 below:


DHIS 2 - Go.Data Interoperability app

Overview
Go.Data Server Credentials
DHIS2 Server Credentials
Metadata Mapping
Interoperability Tasks

Overview: Configure Go.Data-DHIS2 Interoperability App

Configure your settings for sending and receiving data across Go.Data and DHIS2 platforms.

Go.Data Configuration

Add and manage Go.Data server credentials for sending and receiving to DHIS2.

Set up Go.Data server instance

DHIS2 Configuration

Add and manage DHIS2 server credentials for sending and receiving to Go.Data.

Set up DHIS2 server instance

Metadata Mapping

Specify granular metadata mappings for your data exchange – i.e. dataElements and Program Attributes that should map to various Go.Data variables and vice versa.

Map Metadata

Interoperability Tasks

Add and manage tasks for data exchange, i.e. creating outbreaks, mapping locations, sending and receiving cases, sending and receiving contacts, etc.

Manage Tasks

Figure 2. View of module menu system

Main page of the app shows dashboard of menu options providing brief explanation of menu option.

Servers configuration

Server configuration menus are used to collect and store authentication credentials for both: receiving and sending apps. Server configuration menu has three input fields:

Fully qualified URI of the server

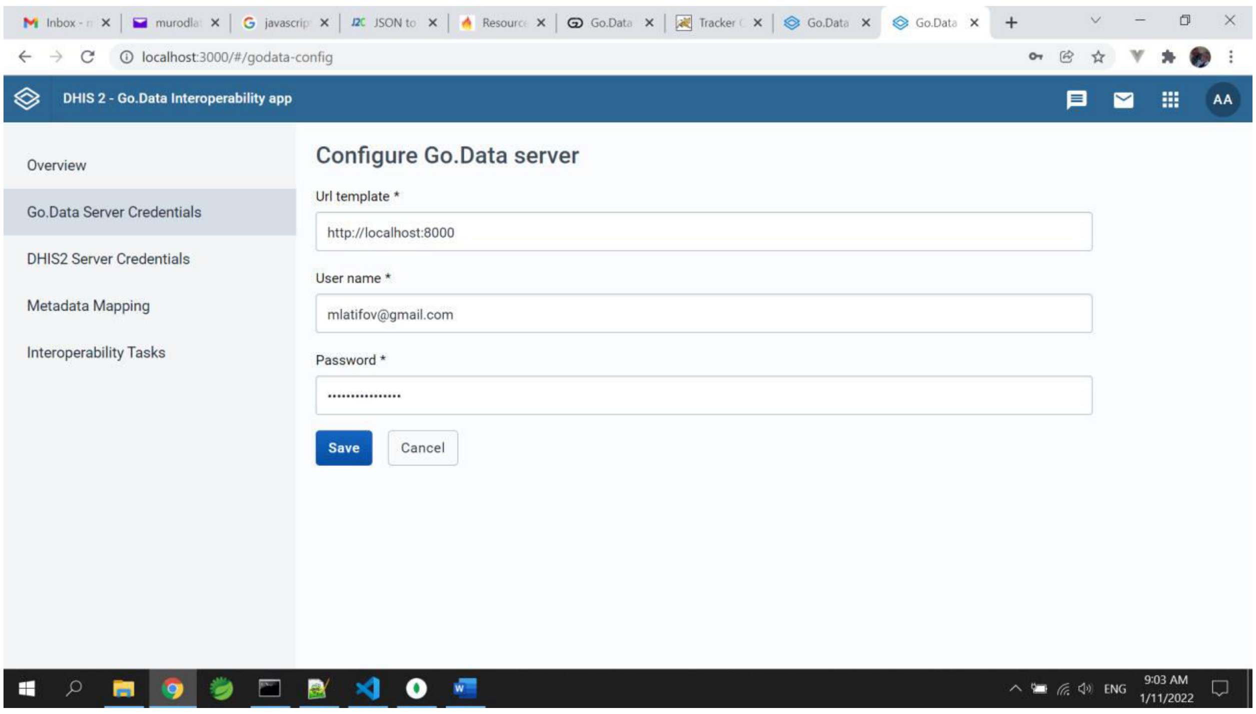
UserName

Password

These data are used to establish secure connection and access data in this server. There are other modes of authorization that need to be counted for, if module needs to be generic and be able to connect to other apps with different mode of authorization.

Go.Data REST API is accessed by providing token that was generated upon submission of username/password. Go.Data token expires quickly, its TTL (timer to live) is short and thus system makes additional calls each time to obtain valid token to GET or POST data.

DHIS2 uses username/password pairs on each call to authorize user. This is called basic authorization, where username and password encoded with base64 before sending. Both authorization types require secure networks.


DHIS 2 - Go.Data Interoperability app

Configure Go.Data server

Url template *: http://localhost:8000

User name *: mlatifov@gmail.com

Password *: ....

Metadata mappings

With provided authorization, app will have access to both apps it provides connectivity for interoperability. Metadata configuration is used to create/modify/delete metadata mappings. For mapping of both API endpoints dot notation is used. Dot notation is a way of accessing JSON objects of any depth following Object.propname.propname…. This technique is used throughout of system development. First system reads Go.Data object instance via API call after successful login. Following this, an instance of received object is converted into array of object, containing the following elements, like:

```
{
        "godata": "parentLocationId"
        "dhis2": "parent.id"
        "props": {
                "conversion": "true"
                "values": { }
        }
},
```

This object has pair of instance element identifiers for both endpoints. In this example, we matched Go.Data “parentLocationId” from Location object to the “parentId” of DHIS2 Organisation Unit object. Next is object “props”, which provides metadefinition to the app on how to process this mapping.

Element “conversion” can have various values, which are keys to navigate conversion process. “values” element provides semantic conversion, if source or destination have coded values. The following example shows mapping of Go.Data Location "geographicalLevelId" to the DHIS2 “level”:

```
{
"godata": "geographicalLevelId"
"dhis2": "level"
"props": {
        "conversion": "true"
        "values":{
                "Admin Level 1": "2"
                "Admin Level 0": "1"
                "Admin Level 2": "3"
                "Admin Level 3": "4"
                "Admin Level 4": "5"
                }
        }
},
```

In both examples above we have conversion value set to true. This signals the system to process ordinary conversion, meaning reading value of DHIS2 “level” and setting it to Go.Data “geographicalLevelId”. But this is codified element, thus we need to convert content as well so its meaningful to the receiving end. In this example if “level” value is 2, Go.Data instance will receive “Admin Level 1” instead of 2.

There are many cases that corresponding element is not provided by the parties. In this case value of “conversion” would be false as we provide value manually. Here is an example of assigning Outbreak Id for processing Cases and Contacts. DHIS2 does not provide this value, we have to add it manually as:

```
{
  "godata":"outbreakId",
  "dhis2":"dbe6a171-3eb3-43ad-b4dd-fe9222bafcb2",
  "props":{
    "conversion":"false",
    "values":{
    }
  }
```

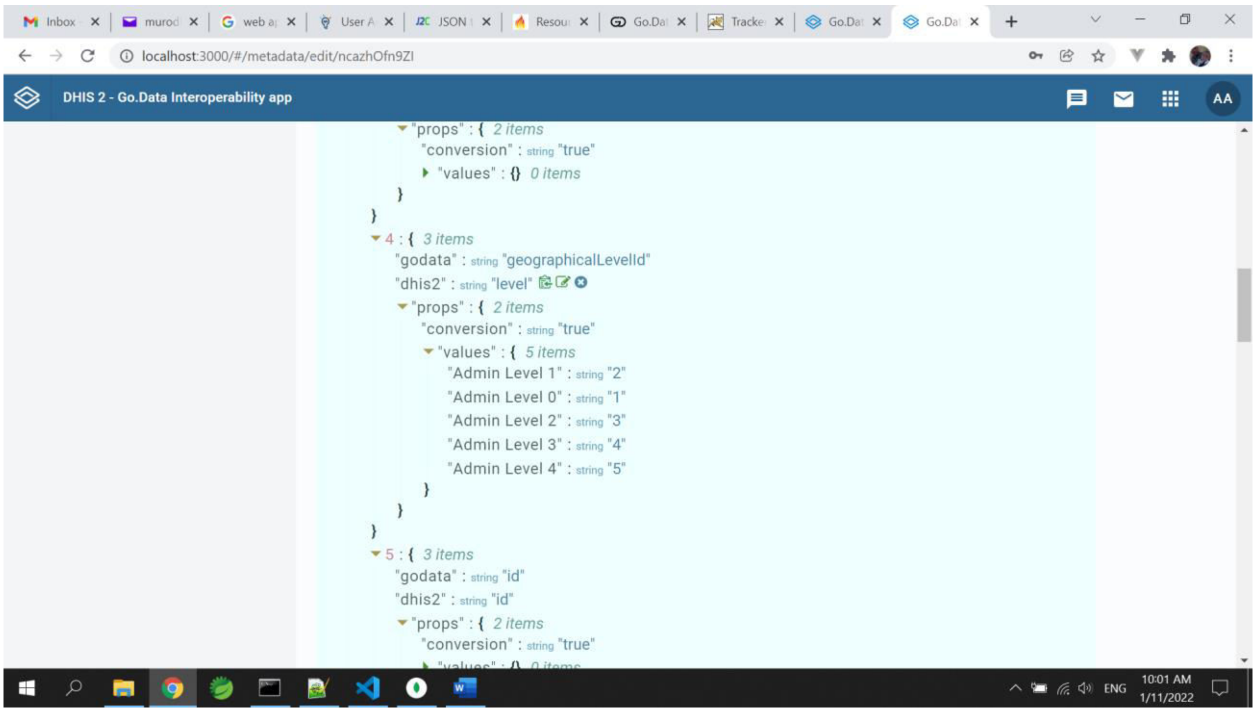
},

There are other cases for conversion and also data for some API endpoints are nested. For example, in Go.Data Cases and Contacts belong to Outbreaks. In order to process Cases and Contacts, one must know id of Outbreak to which these instances should map to. Value of conversion element is set to false as no conversion is needed. We will look at this at next chapter.

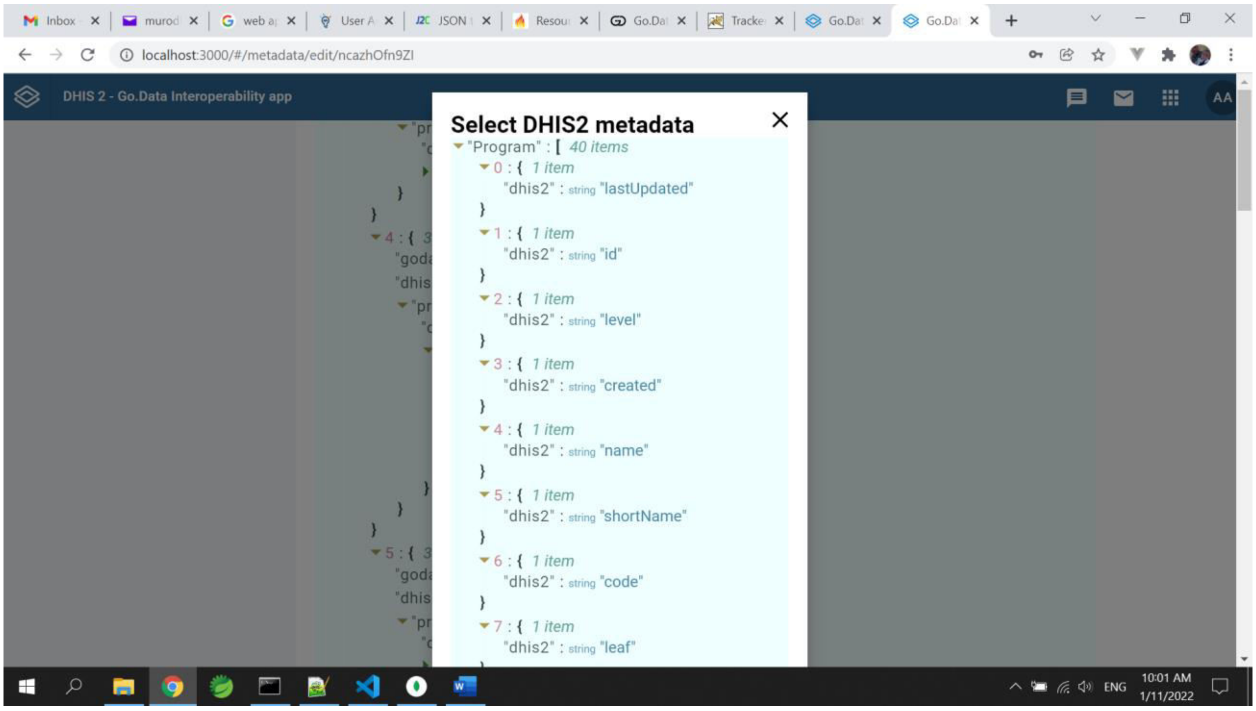
There are currently three specific conversion values: “attr”, “delm” and “geo”. Each of these values signal apps processing signal to process conversion in a specific way. “geo” is used to convert DHIS2 geospatial representation into acceptable georepresentation of Go.Data, which is (longitude, latitude). “geo” will do its best to convert any geotype of DHIS2 (Point, Polygon, Multi Polygon) into Go.Data geotype (Point).

“attr” value of conversion element is used to process DHIS2 Program attributes into Go.Data element values. DHIS2 Programs are highly customizable and can have arbitrary number of Program attribute definitions and there is no standard way to access these attributes other than their UIDs (id). “attr” conversion is used for this specific cases. Same applies to “delm” value for processing DHIS2 data elements.

For the user interface React Json Viewer component is used with slight modifications. Initially pair values JSON is presented with commands. Each command in this editor has specific role of select/add/delete/edit. Select button opens modal with DHIS2 instance of the corresponding Go.Data object instance. While user selects corresponding DHIS2 attribute, it will be linked to pair elements of intermediary JSON object.


localhost:3000/#/metadata/edit/ncazhOfn9ZI
DHIS 2 - Go.Data Interoperability app
AA
10:01 AM
1/11/2022

```
        "props" : { 2 items
            "conversion" : string "true"
            "values" : {} 0 items
        }
    }
    4 : { 3 items
        "godata" : string "geographicalLevelId"
        "dhis2" : string "level"
        "props" : { 2 items
            "conversion" : string "true"
            "values" : { 5 items
                "Admin Level 1" : string "2"
                "Admin Level 0" : string "1"
                "Admin Level 2" : string "3"
                "Admin Level 3" : string "4"
                "Admin Level 4" : string "5"
            }
        }
    }
    5 : { 3 items
        "godata" : string "id"
        "dhis2" : string "id"
        "props" : { 2 items
            "conversion" : string "true"
```

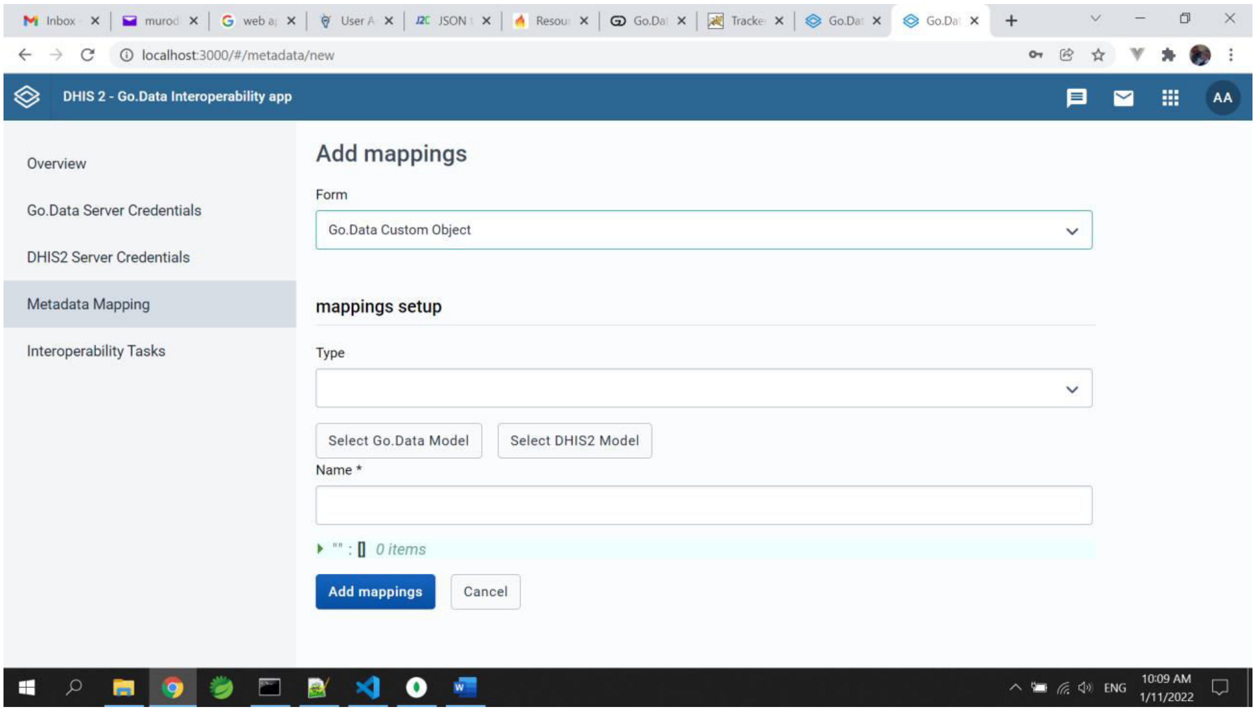


Select DHIS2 metadata

```
"Program" : [ 40 items
  0 : { 1 item
    "dhis2" : string "lastUpdated"
  }
  1 : { 1 item
    "dhis2" : string "id"
  }
  2 : { 1 item
    "dhis2" : string "level"
  }
  3 : { 1 item
    "dhis2" : string "created"
  }
  4 : { 1 item
    "dhis2" : string "name"
  }
  5 : { 1 item
    "dhis2" : string "shortName"
  }
  6 : { 1 item
    "dhis2" : string "code"
  }
  7 : { 1 item
    "dhis2" : string "leaf"
```

Once mapping is finished, it is saved at DHIS2 database as we need it during task running.

Mappings supported at present are: Outbreaks, Locations, Cases, Contacts. There is also possibility to manually adding mapping for manually given endpoints. This is one time shop, meaning after its created no editing is possible. One has to drop and recreate mapping.



localhost:3000/#/metadata/new
DHIS 2 - Go.Data Interoperability app
Overview
Go.Data Server Credentials
DHIS2 Server Credentials
Metadata Mapping
Interoperability Tasks

Add mappings

Form: Go.Data Custom Object

mappings setup

Type:

Select Go.Data Model | Select DHIS2 Model

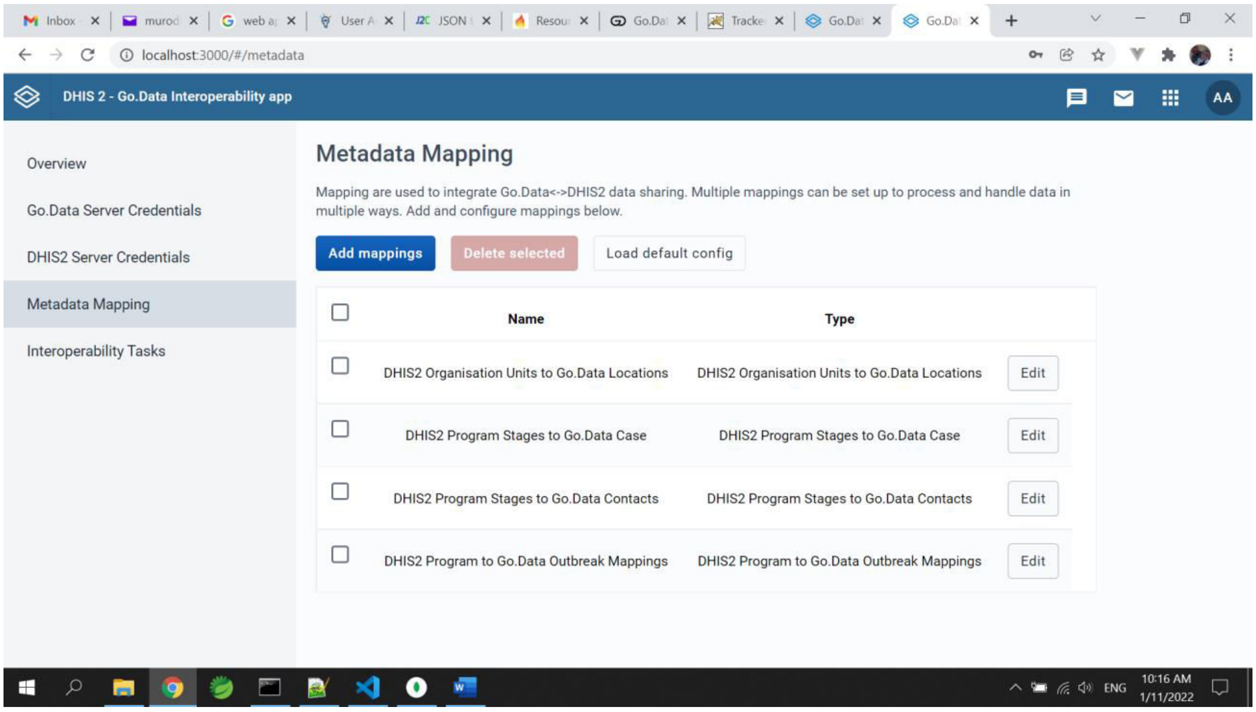
Name *

"" : [] 0 items

Add mappings | Cancel

Custom mapping has two additional inputs: payload for receiving and sending endpoints.

Many of mappings are dynamic in terms that DHIS2 Programs are highly customizable and do not have permanent structure for one time mapping. It is case based mapping and app supports this approach. At present system reads instances of existing Programs and Outbreaks to provide payloads for mapping. In case of Go.Data this could be provided via API calls to metadata, but in DHIS2 it likely to remain the same way due to its ever changing design of Programs. But still two options are more static. These are Organisation Units to Locations mapping, and Program to Outbreak mapping. To ease creation of mapping for Locations and Outbreaks default mapping (sketch) is provided that might need minor editing on new installation of the module. There is a button in “Metadata mapping” menu screen called “Load default config”, which created two mapping instances for Locations and Outbreaks.



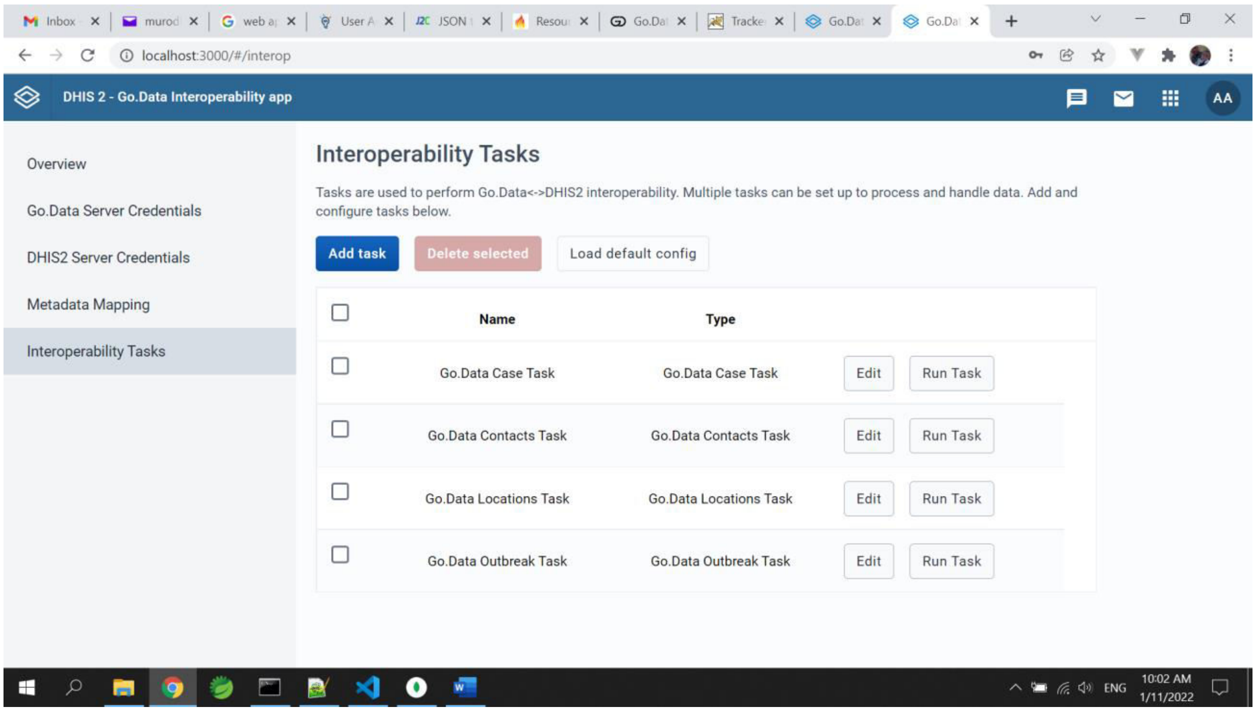
Metadata Mapping

Mapping are used to integrate Go.Data<->DHIS2 data sharing. Multiple mappings can be set up to process and handle data in multiple ways. Add and configure mappings below.

Name	Type	
DHIS2 Organisation Units to Go.Data Locations	DHIS2 Organisation Units to Go.Data Locations	Edit
DHIS2 Program Stages to Go.Data Case	DHIS2 Program Stages to Go.Data Case	Edit
DHIS2 Program Stages to Go.Data Contacts	DHIS2 Program Stages to Go.Data Contacts	Edit
DHIS2 Program to Go.Data Outbreak Mappings	DHIS2 Program to Go.Data Outbreak Mappings	Edit

Interoperability tasks

Fourth menu “Interoperability tasks” is dedicated for managing and running the tasks. Listing of Tasks of this menu has two buttons: Edit and Run Task.

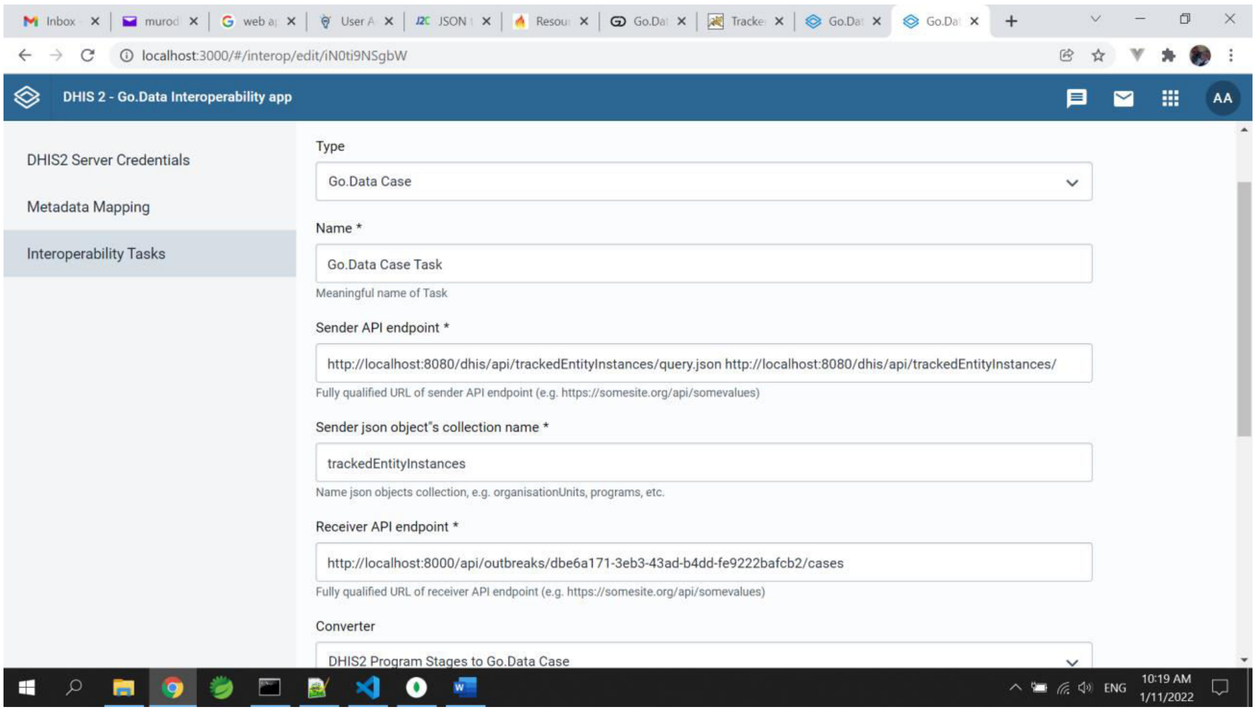


Interoperability Tasks

Tasks are used to perform Go.Data<->DHIS2 interoperability. Multiple tasks can be set up to process and handle data. Add and configure tasks below.

Name	Type		
Go.Data Case Task	Go.Data Case Task	Edit	Run Task
Go.Data Contacts Task	Go.Data Contacts Task	Edit	Run Task
Go.Data Locations Task	Go.Data Locations Task	Edit	Run Task
Go.Data Outbreak Task	Go.Data Outbreak Task	Edit	Run Task

Edit is used to modify existing Task. Task are composed of “name”, “Type”, “senderAPIEndpoint”, “receiverAPIEndpoint”, “Sender json object"s collection name”, “converter”, “referenceModel”, “senderAPIParams” and “sender” elements.



DHIS 2 - Go.Data Interoperability app

localhost:3000/#/interop/edit/iN0ti9NSgbW

- DHIS2 Server Credentials
- Metadata Mapping
- Interoperability Tasks

Type: Go.Data Case

Name *: Go.Data Case Task
Meaningful name of Task

Sender API endpoint *: http://localhost:8080/dhis/api/trackedEntityInstances/query.json http://localhost:8080/dhis/api/trackedEntityInstances/
Fully qualified URL of sender API endpoint (e.g. https://somesite.org/api/somevalues)

Sender json object"s collection name *: trackedEntityInstances
Name json objects collection, e.g. organisationUnits, programs, etc.

Receiver API endpoint *: http://localhost:8000/api/outbreaks/dbe6a171-3eb3-43ad-b4dd-fe9222bafcb2/cases
Fully qualified URL of receiver API endpoint (e.g. https://somesite.org/api/somevalues)

Converter: DHIS2 Program Stages to Go.Data Case

10:19 AM 1/11/2022

“name” is the name of the task

“Type” – type of objects in the conversion (Outbreak, Case, Contact, ContactsOfContact, Location)

“senderAPIEndpoint” API endpoint of sender (http://godata.org/locations)

“Sender json object"s collection name” is the name of root object in response

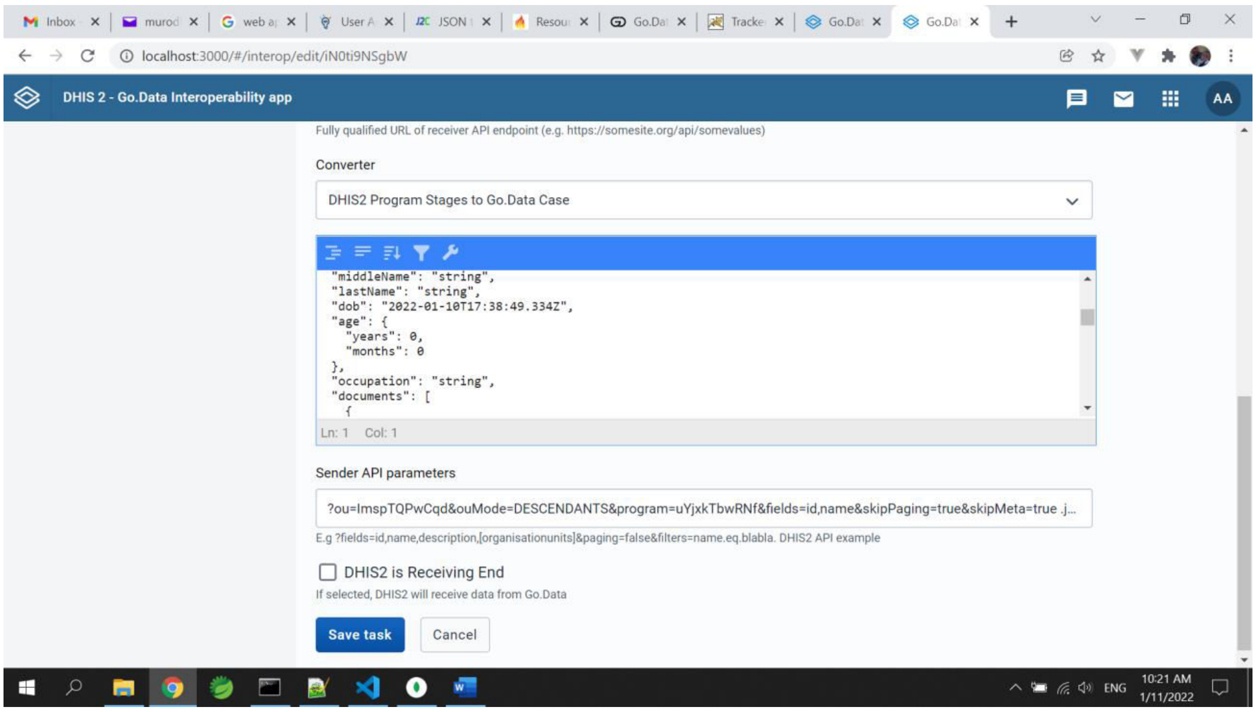
“receiverAPIEndpoint” API endpoint of sender (http://dhis2.org/locations)

“converter” – mapping created in mapping menu for conversion “godata” – “dhis2”

“referenceModel” – JSON model accepted by the receiving endpoint

“senderAPIParams” – parameters to be send to sender endpoint, filters=created.eq.”25/11/2021”&paging=false

“sender” – determines which instance is sender (true – DHIS2, false – Go.Data)



localhost:3000/#/interop/edit/iN0ti9NSgbW

DHIS 2 - Go.Data Interoperability app

Fully qualified URL of receiver API endpoint (e.g. https://somesite.org/api/somevalues)

Converter

DHIS2 Program Stages to Go.Data Case

```
"middleName": "string",
"lastName": "string",
"dob": "2022-01-10T17:38:49.334Z",
"age": {
  "years": 0,
  "months": 0
},
"occupation": "string",
"documents": [
  {
```

Ln: 1 Col: 1

Sender API parameters

?ou=ImspTQPwCqd&ouMode=DESCENDANTS&program=uYjxkTbwRNf&fields=id,name&skipPaging=true&skipMeta=true .j...

E.g ?fields=id,name,description,[organisationunits]&paging=false&filters=name.eq.blabla. DHIS2 API example

DHIS2 is Receiving End

If selected, DHIS2 will receive data from Go.Data

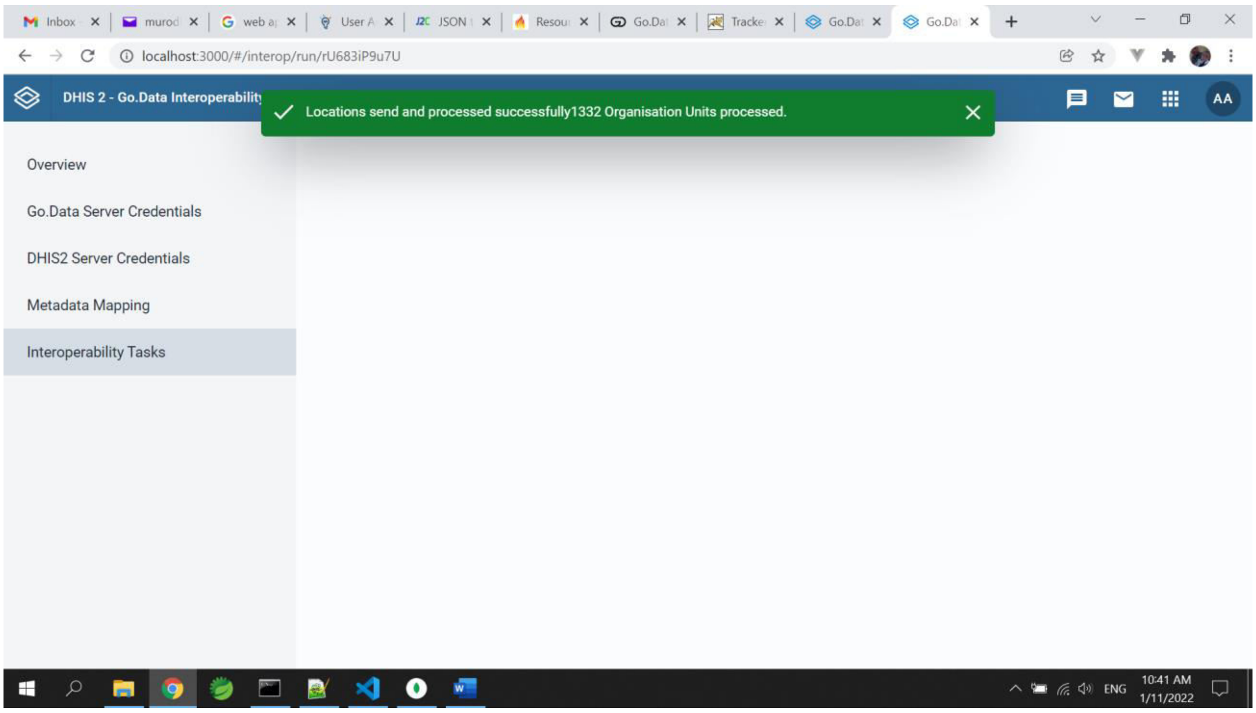
Save task | Cancel

Figure 4.Taks screen

Here special attention should be given to the endpoints of REST APIs. In some cases there is a need for nested calls. At present system can process two consequent calls. For this type of calls endpoints are entered to the related field with space in between. For instance to receive Cases from DHIS2 Program, one need to know certain parameters which are used in the next call.

Two predefined Tasks are created and can be installed by clicking “Load default config” button.

Run button starts processing data exchange. Run process produces information to the user upon each step it performs. Success or Errors are reported on the screen to the user. User uses these information as a confirmation or takes measures if error was reported.



localhost:3000/#/interop/run/rU683iP9u7U

DHIS 2 - Go.Data Interoperability

Locations send and processed successfully1332 Organisation Units processed.

App could be installed and configured in any DHIS2 instance.

Prepared by: Murodillo Latifov, PhD, ICT Expert

25/12/2021,

Dushanbe, Tajikistan