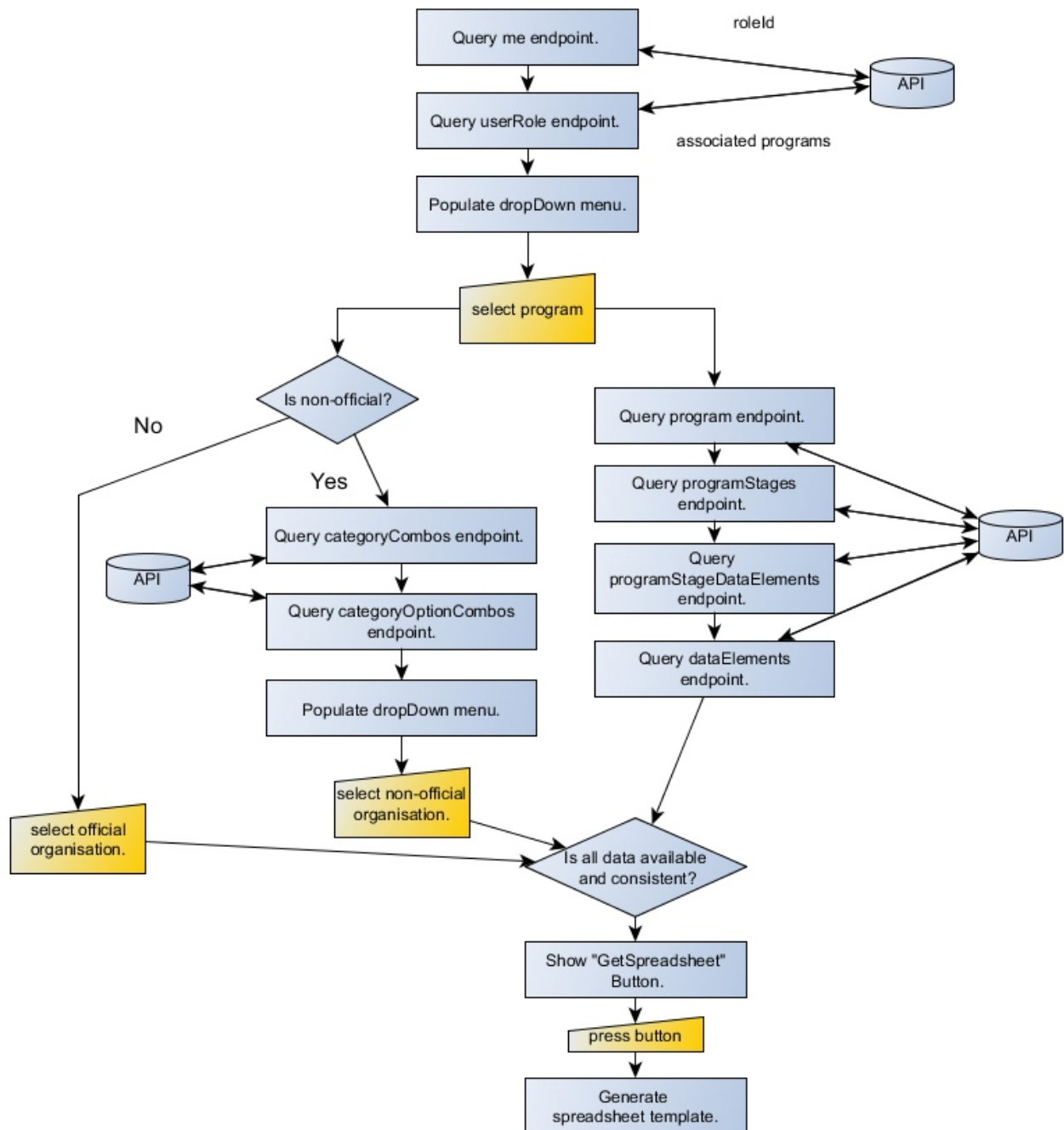


Documentation WISC DHIS2 Bulk Data Upload App



Jakob L.K. Gerstenlauer
UPC Barcelona

Contents

1	Introduction	3
1.1	Motivation	3
1.2	Process Overview	3
2	Workflow	5
2.1	Downloading a Template for Programs	5
2.2	Downloading a Template for Data Sets	5
2.3	Entering Data	6
2.4	Data Upload for Program Data	7
2.5	Data Upload for Data Sets	7
3	Internal Checks and Data Cleaning	8
3.1	Consistency checks	8
3.2	Data cleaning	9
3.3	Differences between Programs and Data Sets	9
4	Acknowledgments	10

1 INTRODUCTION

1.1 MOTIVATION

The World Health Organisation (WHO) has set up monitoring programs for a number of neglected tropical diseases, such as Chagas, because the pharmaceutical industry is not financially motivated to develop drugs and vaccines for these diseases. The Chagas disease is caused by the protozoan *Trypanosoma cruzi* [JRM10]. Several bug species from the *Triatominae* family (from the *Hemiptera* order of so-called “true bugs”) are possible vectors of the disease. However, blood transfusion and contaminated food can also transmit the disease [WHO17]. The WHO has set up the World Information System to control the Chagas Disease (WISCC) using the District Health Information System, an integrated web-based information system that was specifically developed for deployment in developing countries [Tea17]. The Bulk Data Upload App was developed in order to facilitate the bulk upload of data to the DHIS2 data base.

1.2 PROCESS OVERVIEW

In a typical usage scenario, a user first downloads a template spreadsheet, copies the data from another spreadsheet or text file to the template, and finally uploads the completed spreadsheet to the DHIS2 system. In the first step, when downloading the template spreadsheet, the user has to decide if he wants to upload data to a program or to a data set and select from either the drop-down list of data sets or programs. In either case, at least one organizational unit has to be selected from the organizational unit tree. When uploading data to data sets, data from several units can be uploaded from one spreadsheet. In general, the user can only select programs or data sets for which he is authorized. After selecting a data set or program, subsequent drop down lists may open: When selecting a program that is associated with non-official organizations (i.e. NGOs, all programs with UCR** as identifier), the respective data provider has to be additionally selected. When selecting a data set, it might be necessary to choose from a list of categories (i.e. different drug companies providing data to the WHO). The user also has to select the first period for which records should be generated in the template sheet. Depending on the reporting period of data sets, this might involve choosing a year, year and a month, or a year and a calendar week. Once all necessary options have been selected, an excel spreadsheet will be generated with the “Get spreadsheet template” button. The generated spreadsheet template contains four sheets: The first sheet contains a short description of the app and the spreadsheet. The second sheet contains a header with column labels. The user is supposed to enter his data into this sheet. The third sheet contains a legend that describes the meaning, data types, and possible values of the different columns. The last sheet contains meta data which is needed for the data upload. Once, the user has entered his data into the second sheet, he can continue and upload the spreadsheet. If he does not wish to go through

the complete process in one session, he again has to select his organizational unit from the unit tree, the program (or data set), and if applicable his NGO. At the beginning of the upload process, the app tests if the meta data (in sheet nr. 4) coincides with the user supplied information. Then the data is processed and – if no invalid data is detected – the data are send to the DHIS web API¹ and uploaded. The data upload itself is separated into two stages: First, the data is sent in a dry-run mode to the web API. If the API responds that the upload is possible for all supplied data elements, then the data is again sent, and this time it is really imported into the DHIS2 data base. In the following, I will give more details on this workflow.

¹API stands for Application Programming Interface and it represents a query endpoint from which information from the DHIS2 system can be retrieved by apps. DHIS2 provides a number of specialised APIs.

2 WORKFLOW

2.1 DOWNLOADING A TEMPLATE FOR PROGRAMS

The user has to select a program (i.e. form) from a drop-down list. He can only select programs for which he is authorized. The App retrieves this information based on nested calls to the *me* and *userRoles* API. If the program is associated with non-governmental organisations (NGO) – this information can be deduced from the program label – a new drop-down list with the respective NGOs is populated. The organizational unit (short orgunit) has to be selected from an orgunit tree. The template spreadsheet can only be downloaded, if both the orgunit and the program (and if applicable also the NGO) have been selected.

2.2 DOWNLOADING A TEMPLATE FOR DATA SETS

The user has to select a data set from a drop-down list. He can only select data sets for which he is authorized. The App retrieves this information based on nested calls to the *me* and *userRoles* API. A new drop-down list with different data category options for the data set is populated. The user also has to select the first period for which records should be generated in the template sheet. Depending on the reporting period of the data set, this might involve choosing the year, year and month, or year and calendar week. Then, the user can choose several organizational units from the orgunit tree (at least one orgunit has to be selected). Finally, the template spreadsheet can be downloaded.

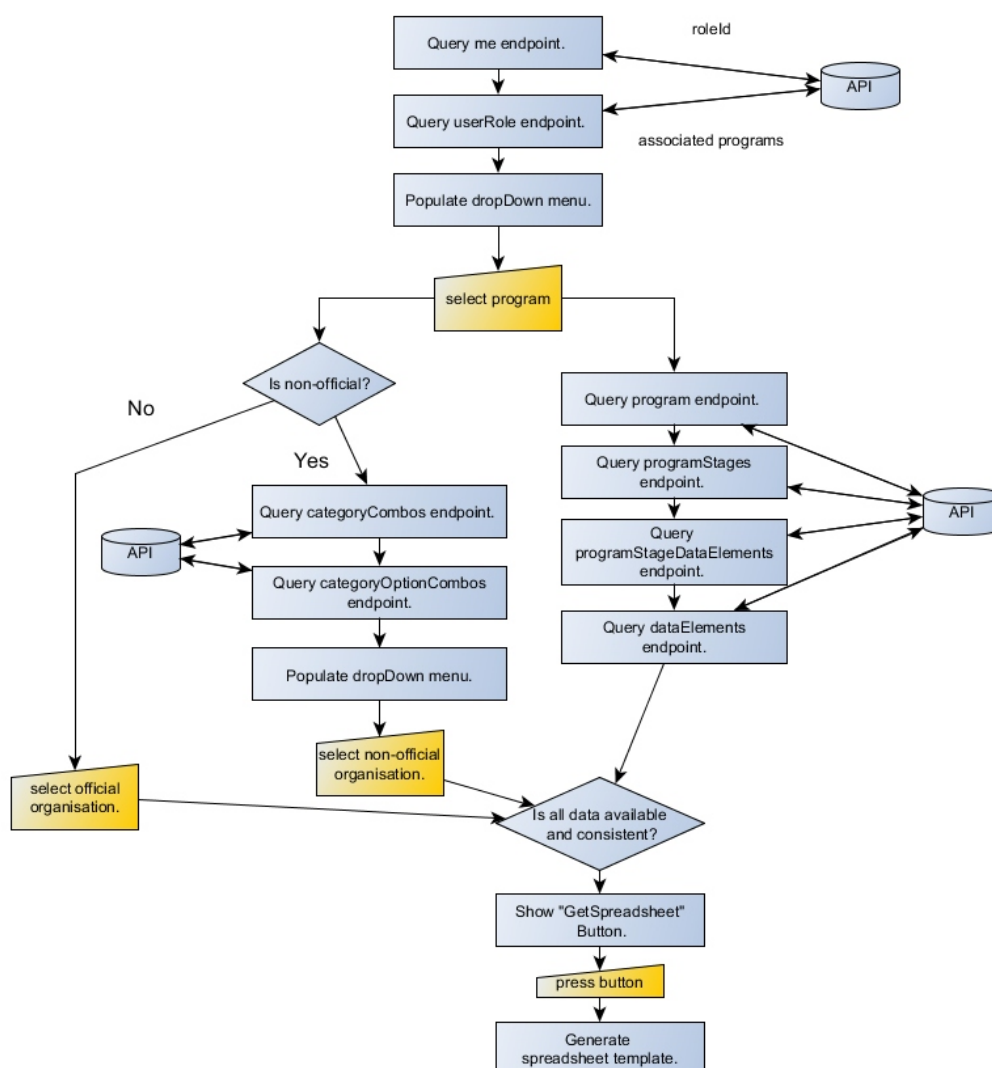


FIGURE 1: DYNAMIC GENERATION OF DROP-DOWN LISTS.

2.3 ENTERING DATA

The generated Excel spreadsheet template contains four sheets:

- The first sheet contains a short description of the app and the spreadsheet structure.
- The second sheet contains a header with column labels.
- The third sheet contains a legend that describes the meaning and data types of the different columns.
- The last sheet contains meta data which is needed for the data upload.

The user is supposed to enter his data into the second sheet. Before, he should have a look at the third sheet which describes the data format of the different columns. This sheet also details which values are possible if the data element is associated with an option set. The last

sheet should not be manipulated by the user, it contains meta data which is necessary for the successful upload of the event data from the spreadsheet. Once, the user has entered his data into the first sheet, he can continue with the upload of the spreadsheet.

2.4 DATA UPLOAD FOR PROGRAM DATA

First, the user has to browse the local file system and select the completed spreadsheet using the **File Selection** button. Next, he can choose if the app should validate the geographical location data. For expert users, it is also possible to choose between different log levels in the central panel. The **Upload** button triggers the data upload. If the upload succeeds, a summary of the number of uploaded rows will be printed to the text window. Else, error messages will hint to the potential error sources. During the upload step, the app runs several data cleaning and consistency checks which will be described in the next section.

2.5 DATA UPLOAD FOR DATA SETS

First, the user has to browse the local file system and select the completed spreadsheet using the **File Selection** button. Next, he can choose the “import strategy”:

- **CREATE:** Create new data entries and ignore new values for existing entries.
- **UPDATE:** Update the values of existing entries.
- **CREATE AND UPDATE:** Create new data entries and update values for existing entries.
- **DELETE:** Delete the supplied data entries.

3 INTERNAL CHECKS AND DATA CLEANING

3.1 CONSISTENCY CHECKS

By selecting the option “Latitude and longitude are provided”, the user forces the app to validate the geographical location data. By default the app does not validate the location data. In the following it is assumed that the user chose to validate the location data.

At the beginning of the upload process, the app tests if the currently selected IDs of program and organizational unit coincide with the meta data which is stored in the third sheet of the Excel file. If any of the IDs do not match, the process is aborted with an error message. In this way, the app ensures that users which submit data to different programs do not try to send data to the wrong program. Users also have to select the correct organizational level in the organization unit tree.

Before the rows are processed, rows are checked for duplicates using the md5 hash function. If there is any duplicate row, the upload process is aborted with an error message. If all these checks were successful, the app proceeds with the following row-wise data checks:

- Are values supplied for the first three columns “ReportingDate”, “Latitude”, “Longitude”?
- Is the “ReportingDate” correctly supplied with a string of 23 characters?
- Is the “Latitude” value a valid numeric value between -90.0 and +90.0?
- Is the “Longitude” value a valid numeric value between -180.0 and +180.0?
- Is this a valid geographical location for the polygon² of the organisational unit?
- Is there at least one value for an optional³ data element?
- Is the supplied value within the set of permitted values of the option set?

Although users can provide rows with missing data, they have to provide all three mandatory columns (compare box) and at least one additional data element. The upload of the data set as a whole does not start before all input rows have been processed. If any of these checks fails for any line, the upload process will be aborted once all rows have been read. The rational is that users should be informed about all possible errors before aborting the process.

²The polygon of the organisational unit is recursively retrieved from the *organisationUnits* API: If no polygon can be retrieved for the unit, I retry the query with the parent unit. A maximum recursion depth of three is used. If no polygon is retrievable for the grandparent unit, a warning message is printed informing the user that the location could not be validated because of missing polygon information.

³Optional elements are all data elements except “ReportingDate”, “Latitude”, and “Longitude”.

FRAME 1: MANDATORY DATA ELEMENTS

- ReportingDate: 2016-12-01T00:00:00.000
- Latitude $\in [-90.0, +90.0]$
- Longitude $\in [-180.0, +180.0]$
- (Latitude,Longitude) \in Polygon

3.2 DATA CLEANING

Based on a query to the *dataElements* API, the properties *formName* and *valueType* are available for all data elements: The property *formName* is a informative label that is used as header information in the first spreadsheet. The property *valueType* describes the data type and it is used to infer the appropriate type of data cleaning operations:

TABLE 1: VALUE TYPE DEPENDENT DATA CLEANING.

Value type	Remove quotes	Further actions
"COORDINATE"	Yes	
"LONG_TEXT"	Yes	
"TEXT"	Yes	
"INTEGER_POSITIVE"	No	if($x \leq 0$): "", else: x
"TRUE_ONLY"	Yes	if(regex("TRUE")): "true", else: ""

3.3 DIFFERENCES BETWEEN PROGRAMS AND DATA SETS

There are a few minor differences in the consistency checks for data that is uploaded to programs and data sets. For data sets, the geographical location is not checked and it is possible to supply values for several organizational units. The data cleaning and the check of valid options for option sets are identical.

4 ACKNOWLEDGMENTS

I am very thankful to Eric Mourin Marin who introduced me to DHIS2 and often helped me in the practical implementation of the app. The developers of the DHIS2 Excel Data Upload App did a great job developing their app and we used some of their code to implement the current app. Marc Garnica Caparros shared his experience using the Excel Data Upload App and some customized Javascript code with me. Petar Jovanovic was defining the desired features of the app and was always available when I had questions or doubts. Lise Grout, Alberto Abelló, Pedro Albajar Vinas, and Oscar Romero Moral were defining the general requirements of the app.

References

ARTICLES

- [JRM10] Anis Rassi Jr, Anis Rassi, and José Antonio Marin-Neto. “Chagas disease”. In: *The Lancet* 375.9723 (2010), pp. 1388–1402. ISSN: 0140-6736. DOI: [http://dx.doi.org/10.1016/S0140-6736\(10\)60061-X](http://dx.doi.org/10.1016/S0140-6736(10)60061-X). URL: <http://www.sciencedirect.com/science/article/pii/S014067361060061X>.

ONLINE

- [Tea17] DHIS2 Documentation Team. *DHIS2 Implementation Guide Version 2.27*. 2017. URL: https://docs.dhis2.org/2.24/en/implementer/dhis2_implementation_guide.pdf.
- [WHO17] WHO. *Chagas disease (American trypanosomiasis)*. 2017. URL: <http://www.who.int/mediacentre/factsheets/fs340/en>.