DHIS2 Implementer guide

2.31

DHIS2 Documentation team

2018

1 About this guide

The DHIS2 documentation is a collective effort and has been developed by the development team and users. While the guide strives to be complete, there may be certain functionalities which have been omitted or which have yet to be documented. This section explains some of the conventions which are used throughout the document.

DHIS2 is a browser-based application. In many cases, screenshots have been included for enhanced clarity. Shortcuts to various functionalities are displayed such as **Data element** > **Data element group**. The ">" symbol indicates that you should click **Data element** and then click **Data element group** in the user interface.

Different styles of text have been used to highlight important parts of the text or particular types of text, such as source code. Each of the conventions used in the document are explained below.

> **Note**
>
> A note contains additional information which should be considered or a reference to more information which may be helpful.

> **Tip**
>
> A tip can be a useful piece of advice, such as how to perform a particular task more efficiently.

> **Important**
>
> Important information should not be ignored, and usually indicates something which is required by the application.

> **Caution**
>
> Information contained in these sections should be carefully considered, and if not heeded, could result in unexpected results in analysis, performance, or functionality.

> **Warning**
>
> Information contained in these sections, if not heeded, could result in permanent data loss or affect the overall usability of the system.

```
Program listings usually contain some type of computer code.
They will be displayed with a shaded background and a different font.
```

`Commands will be displayed in bold text, and represent a command which would need to be executed on the operating system or database.`

Links to external web sites or cross references will be displayed in blue text, and underlined like this..

Bibliographic references will displayed in square brackets like this Store2007. A full reference can be found in the bibliography contained at the end of this document.

2 A quick guide to DHIS2 implementation

Any implementation of District Health Information Software (DHIS2) should aim at establishing sustainable systems that are flexible to changing needs in the health sector. It is important to acknowledge that this will take many years, with continuous structures for capacity building, best practice sharing, and innovation. This quick guide will provide a very crude overview of the different facets of DHIS2 implementation.

2.1 Planning and organizing

2.1.1 Structures needed

- A DHIS core team (DCT) of 4-5 people will be needed to administer a national HMIS. Their responsibilities and required skills should be clearly defined. The DCT will participate in DHIS2 Academies, organize training and end-user support for various user groups in the country.
- A Technical Steering Committee, or equivalent, will be needed to steer the coordination between health programs, other information systems and development partners and Universities. They will lead integration efforts and make decisions regarding the overall architecture of information systems.

2.1.2 Integration efforts

- Throughout the implementation, simultaneous efforts of information system integration and data exchange need to be conducted. The leading principle for this work should be to create a decision-driven and indicator-focused system.

2.1.3 Equipment and internet

- An assessment needs to establish the needs for hardware. Desktops, laptops, tablets, mobile phones all have different qualities, and typically a mix of these different technologies will need to be supported.
- Server and hosting alternatives needs to be critically examined with regards to capacity, infrastructural constraints, legal framework, security and confidentiality issues.
- Internet connection for all users will be needed. Mobile internet will be adequate for majority of users doing data collection and regular analysis.
- Options for mobile phone users, bulk sms deals etc, should be examined if appropriate.

2.1.4 Roll-out strategy

- The DCT will play a key role here and each member should have clear responsibilities for the roll-out covering: user support, user training, liaison with health programs, etc.
- Broader support structures need to be established to provide support, supervision, and communication with global/regional network of expert users and developers.
- Information use must be a focus area from the start and be a component both in the initial system design and the first round of user training. Data collection and data quality will only increase with real value of the information. District review meetings and equivalent should be supported with appropriate information products and training.
- Training will typically be the largest investment over time, and necessitates structures for continuous opportunities. Plan for a long term training approach catering for a continuous process of enabling new users and new system functionalities.

- Supervision and data quality assessment should be held frequently.

2.2 Adapting DHIS2

2.2.1 Scope of system

- Based on the decisions the system should support (system scope); customization and adaptation of the platform will be needed for DHIS2 aggregate, tracker, and/or events. Each action will need special competence, and should be led by the DCT.
- Assessment of the intended users and beneficiaries is needed, such as related to their information needs, and hardware and network needs.
- An understanding of the larger architecture of the HIS (the "HIS ecosystem") is important; what other systems are there, and how should they interact with DHIS2? Consider what needs there will be for interoperability between electronic systems.
- If there are needs that are not currently supported by DHIS2, an assessment of additional software development is necessary. These can be addressed locally by developing a custom web app or feed into the overall core platform development roadmap process organized by UiO.

2.2.2 Setting up DHIS2

- Reporting units: implementing the different reporting units (service outlets) and hierarchies including grouping.
- Data collection needs: Which indicators are needed, what data variables will go into their calculation, and how should this data be collected? Design data elements, disaggregation categories, data sets, and collection forms.
- Information for action (indicators, dashboards, other outputs): what are the information products the various users will need? Tables, charts, maps, dashboards. Routines for dissemination and sharing.
- User management: Create user roles and groups, routines for managing users, define access to features, and appropriate sharing of content.
- DHIS2 governance document (roles by profile, how to change metadata and under what conditions).

2.2.3 Hosting

- There are many different options for hosting an online system, both in terms of where to put the server (e.g. in-house vs. cloud) and who to manage the server (e.g. in-house vs. outsourced). Server and hosting alternatives needs to be critically examined with regards to capacity, infrastructural constraints, legal framework, security and confidentiality issues. These decisions may need to be revisited at least annually as server complexity, data types (e.g. aggregate vs. patient) and local capacity may change over time.

2.3 Capacity building

2.3.1 DHIS core team (DCT)

- DCT will need all skills necessary for a sustainable, evolving system. This includes technical skills (DHIS2 adaptation, server maintenance), system knowledge (architectures and design principles), organizational (integration strategies), and project management (organizing structured support and training).

- DCT members should attend the regional/global DHIS2 Academy frequently (e.g. twice a year) to ensure high quality training, continuous communication with the broader expert community, and to make sure the local team is up to date with new functionalities and enhancements in recent releases of the DHIS 2 platform. DCT will be responsible for adapting and cascading this regional training curriculum to a broader group of users within country.

2.3.2 Country training strategies

- DCT should offer training in relation to the implementation, and continuously thereafter to meet growing demands, system updates and staff turnover.
- Adapting and developing training material and reference guides to reflect local information needs and local system content is important.

2.3.3 Continuous training opportunities

- As user experience is growing, more advanced training should be offered. Information use training for district medical officers and health program managers is crucial early on to enroll stakeholders to use the information in decision making.

3 Conceptual Design Principles

This chapter provides a introduction to some of the key conceptual design principles behind the DHIS2 software. Understanding and being aware of these principles will help the implementer to make better use of the software when customising a local database. While this chapter introduces the principles, the following chapters will detail out how these are reflected in the database design process.

The following conceptual design principles will be presented in this chapter:

- All meta data can be added and modified through the user interface
- A flexible data model supports different data sources to be integrated in one single data repository
- Data Input != Data Output
- Indicator-driven data analysis and reporting
- Maintain disaggregated facility-data in the database
- Support data analysis at any level in the health system

In the following section each principle is described in more detail.

3.1 All meta data can be added and modified through the user interface

The DHIS2 application comes with a set of generic tools for data collection, validation, reporting and analysis, but the contents of the database, e.g. what data to collect, where the data comes from, and on what format, will depend on the context of use. These meta data need to be populated into the application before it can be used, and this can be done through the user interface and requires no programming. This allows for more direct involvement of the domain experts that understand the details of the HIS that the software will support.

The software separates the key meta data that describes the raw data being stored in the database, which is the critical meta data that should not change much over time (to avoid corrupting the data), and the higher level meta like indicator formulas, validation rules, and groups for aggregation as well as the various layouts for collection forms and reports, which are not that critical and can be changed over time without interfering with the raw data. As this higher level meta data can be added and modified over time without interfering with the raw data, a continuous customisation process is supported. Typically new features are added over time as the local implementation team learn to master more functionality, and the users are gradually pushing for more advanced data analysis and reporting outputs.

3.2 A flexible data model supports different data sources to be integrated in one single data repository

The DHIS2 design follows an integrated approach to HIS, and supports integration of many different data sources into one single database, sometime referred to as an integrated data repository or a data warehouse.

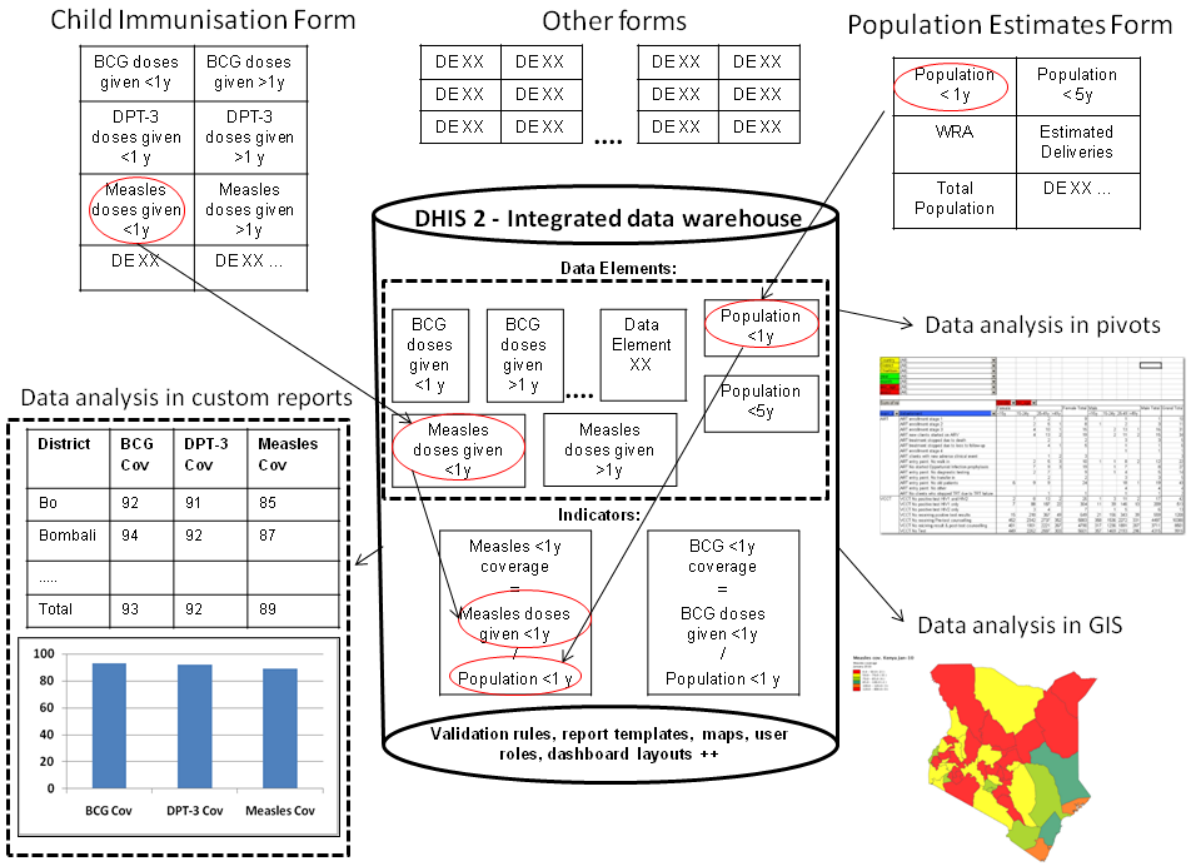
The fact that DHIS2 is a skeleton like tool without predefined forms or reports means that it can support a lot of different aggregate data sources. There is nothing really that limits the use to the health domain either, although use in other sectors are still very limited. As long as the data is collected by an orgunit (organisational unit), described as a data element (possibly with some disaggregation categories), and can be represented by a predefined period frequency, it can be collected and processed in DHIS2. This flexibility makes DHIS2 a powerful tool to set up integrated systems that bring together collection tools, indicators, and reports from multiple

health programs, departments or initiatives. Once the data is defined and then collected or imported into a DHIS2 database, it can be analysed in correlation to any other data in the same database, no matter how and by whom it was collected. In addition to supporting integrated data analysis and reporting, this integrated approach also helps to rationalise data collection and reduce duplication.

3.3 Data input != Data output

In DHIS2 there are three dimensions that describe the aggregated data being collected and stored in the database; the where - organisation unit, the what - data element, and the when - period. The organisation unit, data element and period make up the three core dimensions that are needed to describe any data value in the DHIS2, whether it is in a data collection form, a chart, on a map, or in an aggregated summary report. When data is collected in an electronic data entry form, sometimes through a mirror image of the paper forms used at facility level, each entry field in the form can be described using these three dimensions. The form itself is just a tool to organise the data collection and is not describing the individual data values being collected and stored in the database. Being able to describe each data value independently through a Data Element definition (e.g. 'Measles doses given <1 year') provides important flexibility when processing, validating, and analysing the data, and allows for comparison of data across collection forms and health programs.

This design or data model approach separates DHIS2 from many of the traditional HIS software applications which treat the data collection forms as the key unit of analysis. This is typical for systems tailored to vertical programs' needs and the traditional conceptualisation of the collection form as also being the report or the analysis output. The figure below illustrates how the more fine-grained DHIS2 design built around the concept of Data Elements is different and how the input (data collection) is separated from the output (data analysis), supporting more flexible and varied data analysis and dissemination. The data element 'Measles doses given <1 y' is collected as part of a Child Immunisation collection form, but can be used individually to build up an Indicator (a formula) called 'Measles coverage <1y' where it is combined with the data element called 'Population <1y', being collected through another collection form. This calculated Indicator value can then be used in data analysis in various reporting tools in DHIS2, e.g. custom designed reports with charts, pivot tables, or on a map in the GIS module.



3.4 Indicator-driven data analysis and reporting

What is referred to as a Data Element above, the key dimension that describes what is being collected, is sometimes referred to as an indicator in other settings. In DHIS2 we distinguish between Data Elements which describe the raw data, e.g. the counts being collected, and Indicators, which are formula-based and describe calculated values, e.g. coverage or incidence rates that are used for data analysis. Indicator values are not collected like the data (element) values, but instead calculated by the application based on formulas defined by the users. These formulas are made up of a factor (e.g. 1, 100, 100, 100 000), a numerator and a denominator, the two latter are both expressions based on one or more data elements. E.g. the indicator “Measles coverage <1 year” is defined a formula with a factor 100, a numerator (“Measles doses given to children under 1 year”) and a denominator (“Target population under 1 year”). The indicator “DPT1 to DPT3 drop out rate” is a formula of 100 % x (“DPT1 doses given”- “DPT3doses given”) / (“DPT1 doses given”). These formulas can be added and edited through the user interface by a user with limited training, as they are quite easy to set up and do not interfere with the data values stored in the database (so adding or modifying an indicator is not a critical operation).

Indicators represent perhaps the most powerful data analysis feature of the DHIS2, and all reporting tools support the use of indicators, e.g. as displayed in the custom report in the figure above. Being able to use population data in the denominator enables comparisons of health performance across geographical areas with different target populations, which is more useful than only looking at the raw numbers. The table below uses both the raw data values (Doses) and indicator values (Cov) for the different vaccines. Comparing e.g. the two first orgunits in the list, Taita Taveta County and Kilifi County, on DPT-1 immunisation, we can see that while the raw numbers (659 vs 2088) indicate many more doses are given in Kilifi, the coverage rates (92.2 % vs 47.5 %) show that Taita Taveta are doing a better job immunising their target population under 1 year. Looking at the final column (Immuniz. Compl. %) which indicates the completeness of reporting of the immunisation form for the same period, we can see that the numbers are more

or less the same in the two counties we compared, which tells us that the coverage rates can be reasonably compared across the two counties.

	DPT 1		DPT 2		DPT 3		Measles		Fully Imm		Immuniz Compl %
Organisation	Doses	Cov	Doses	Cov	Doses	Cov	Doses	Cov	Doses	Cov	
Taita Taveta County	659	92.2	630	89.1	580	81.0	574	80.2	551	77.4	81.4
Kilifi County	2088	47.5	1767	39.6	1954	43.0	3315	73.4	2540	55.5	82.1
Lamu County	238	82.6	200	70.0	268	91.4	283	97.7	195	60.6	81.0
Kwale County	1142	51.1	1077	48.3	1149	52.4	1909	86.3	1385	62.2	84.1
Mombasa County	2015	73.2	1711	62.4	1948	70.7	2737	98.0	2278	82.2	88.4
Tana River County	678	80.1	633	77.4	721	76.2	656	79.6	404	50.0	78.3
Coast	6820	60.6	6018	53.5	6620	57.7	9474	83.5	7353	64.6	83.2

3.5 Maintain disaggregated facility-data in the database

When data is collected and stored in DHIS2 it will remain disaggregated in the database with the same level of detail as it was collected. This is a major advantage of having a database system for HIS as supposed to a paper-based or even spreadsheet based system. The system is designed to store large amounts of data and always allow drill-downs to the finest level of detail possible, which is only limited by how the data was collected or imported into the DHIS2 database. In a perspective of a national HIS it is desired to keep the data disaggregated by health facility level, which is often the lowest level in the orgunit hierarchy. This can be done even without computerising this level, through a hybrid system of paper and computer. The data can be submitted from health facilities to e.g. district offices by paper (e.g. on monthly summary forms for one specific facility), and then at the district office they enter all the facility data into the DHIS2 through the electronic data collection forms, one facility at a time. This will enable the districts health management teams to perform facility-wise data analysis and to e.g. provide print-outs of feedback reports generated by the DHIS2, incl. facility comparisons, to the facility in-charges in their district.

3.6 Support data analysis at any level in the health system

While the name DHIS2 indicates a focus on the District, the application provides the same tools and functionality to all levels in the health system. In all the reporting tools the users can select which orgunit or orgunit level to analyse and the data displayed will be automatically aggregated up to the selected level. The DHIS2 uses the orgunit hierarchy in aggregating data upwards and provides data by any orgunit in this hierarchy. Most of the reports are run in such a way that the users will be prompted to select an orgunit and thereby enable reuse of the same report layouts for all levels. Or if desired, the report layouts can be tailored to any specific level in the health system if the needs differ between the levels.

In the GIS module the users can analyse data on e.g. the sub-national level and then by clicking on the map (on e.g. a region or province) drill down to the next level, and continue like this all the way down to the source of the data at facility level. Similar drill-down functionality is provided in the Excel Pivot Tables that are linked to the DHIS2 database.

To speed up performance and reduce the response-time when providing aggregated data outputs, which may include many calculations (e.g. adding together 8000 facilities), DHIS2 pre-calculates all the possible aggregate values and stores these in what is called a data mart. This data mart can be scheduled to run (re-built) at a given time interval, e.g. every night.

4 Setting Up a New Database

The DHIS2 application comes with a set of tools for data collection, validation, reporting and analysis, but the contents of the database, e.g. what data to collect, where the data comes from, and on what format will depend on the context of use. These meta data need to be populated into the application before it can be used, and this can be done through the user interface and requires no programming. What is required is in-depth knowledge about the local HIS context as well as an understanding of the DHIS2 design principles (see the chapter “Key conceptual design principles in DHIS2”). We call this initial process database design or customisation. This chapter provides an overview of the customisation process and briefly explains the steps involved, in order to give the implementer a feeling of what this process requires. Other chapters in this manual provide a lot more detail into some of the specific steps.

4.1 Strategies for getting started

The following section describes a list of tips for getting off to a good start when developing a new database.

1. Quickly populate a demo database, incl. examples of reports, charts, dashboard, GIS, data entry forms. Use real data, ideally nation-wide, but not necessarily facility-level data.
2. Put the demo database online. Server hosting with an external provider server can be a solution to speed up the process, even if temporary. This makes a great collaborative platform and dissemination tool to get buy-in from stakeholders.
3. The next phase is a more elaborate database design process. Parts of the demo can be reused if viable.
4. Make sure to have a local team with different skills and background: public health, data administrator, IT and project management.
5. Use the customisation and database design phase as a learning and training process to build local capacity through learning-by-doing.
6. The country national team should drive the database design process but be supported and guided by experienced implementers.

4.2 Controlled or open process?

As the DHIS2 customisation process often is and should be a collaborative process, it is also important to have in mind which parts of the database are more critical than others, e.g. to avoid an untrained user to corrupt the data. Typically it is a lot more critical to customise a database which already has data values, than working with meta data on an “empty” database. Although it might seem strange, much customisation takes place after the first data collection or import has started, e.g. when adding new validation rules, indicators or report layouts. The most critical mistake that can be made is to modify the meta data that directly describes the data values, and these as we have seen above, are the *data elements* and the *organisation units*. When modifying these definitions it is important to think about how the change will affect the meaning of the data values already in the system (collected using the old definitions). It is recommended to limit who can edit these core meta data through the user role management, to restrict the access to a core customisation team.

Other parts of the system that are not directly coupled to the data values are a lot less critical to play around with, and here, at least in the early phases, one should encourage the users to try out new things in order to create learning. This goes for groups, validation rules, indicator formulas, charts, and reports. All these can easily be deleted or modified later without affecting the underlying data values, and therefore are not critical elements in the customisation process.

Of course, later in the customisation process when going into a production phase, one should be even more careful in allowing access to edit the various meta data, as any change, also to the less critical meta data, might affect how data is aggregated together or presented in a report (although the underlying raw data is still safe and correct).

4.3 Steps for developing a database

The following section describes concrete steps for developing a database from scratch.

4.3.1 The organisational hierarchy

The organisational hierarchy defines the organisation using the DHIS2, the health facilities, administrative areas and other geographical areas used in data collection and data analysis. This dimension to the data is defined as a hierarchy with one root unit (e.g. Ministry of Health) and any number of levels and nodes below. Each node in this hierarchy is called an organisational unit in DHIS2. The design of this hierarchy will determine the geographical units of analysis available to the users as data is collected and aggregated in this structure. There can only be one organisational hierarchy at the same time so its structure needs careful consideration.

Additional hierarchies (e.g. parallel administrative boundaries to the health care sector) can be modelled using organisational groups and group sets, but the organisational hierarchy is the main vehicle for data aggregation on the geographical dimension. Typically national organisational hierarchies in public health have 4-6 levels, but any number of levels is supported. The hierarchy is built up of parent-child relations, e.g. a Country or MoH unit (the root) might have e.g. 8 child units (provinces), and each province again (at level 2) might have 10-15 districts as their children. Normally the health facilities will be located at the lowest level, but they can also be located at higher levels, e.g. national or provincial hospitals, so skewed organisational trees are supported (e.g. a leaf node can be positioned at level 2 while most other leaf nodes are at level 5).

4.3.2 Data Elements

The Data Element is perhaps the most important building block of a DHIS2 database. It represents the *what* dimension, it explains what is being collected or analysed. In some contexts this is referred to an indicator, but in DHIS2 we call this unit of collection and analysis a data element. The data element often represents a count of something, and its name describes what is being counted, e.g. “BCG doses given” or “Malaria cases”. When data is collected, validated, analysed, reported or presented it is the data elements or expressions built upon data elements that describes the WHAT of the data. As such the data elements become important for all aspects of the system and they decide not only how data is collected, but more importantly how the data values are represented in the database, which again decides how data can be analysed and presented.

A best practice when designing data elements is to think of data elements as a unit of data analysis and not just as a field in the data collection form. Each data element lives on its own in the database, completely detached from the collection form, and reports and other outputs are based on data elements and expressions/formulas composed of data elements and not the data collection forms. So the data analysis needs should drive the process, and not the look an feel of the data collection forms.

4.3.3 Data sets and data entry forms

All data entry in DHIS2 is organised through the use of data sets. A data set is a collection of data elements grouped together for data collection, and in the case of distributed installs they also define chunks of data for export and import between instances of DHIS2 (e.g. from a district office local installation to a national server). Data sets are not linked directly to the data values,

only through their data elements and frequencies, and as such a data set can be modified, deleted or added at any point in time without affecting the raw data already captured in the system, but such changes will of course affect how new data will be collected.

Once you have assigned a data set to an organisation unit that data set will be made available in Data Entry (under Services) for the organisation units you have assigned it to and for the valid periods according to the data set's period type. A default data entry form will then be shown, which is simply a list of the data elements belonging to the data set together with a column for inputting the values. If your data set contains data elements with categories such as age groups or gender, then additional columns will be automatically generated in the default form based on the categories. In addition to the default list-based data entry form there are two more alternatives, the section-based form and the custom form. Section forms allow for a bit more flexibility when it comes to using tabular forms and are quick and simple to design. Often your data entry form will need multiple tables with subheadings, and sometimes you need to disable (grey out) a few fields in the table (e.g. some categories do not apply to all data elements), both of these functions are supported in section forms. When the form you want to design is too complicated for the default or section forms then your last option is to use a custom form. This takes more time, but gives you full flexibility in term of the design. In DHIS2 there is a built in HTML editor (FcK Editor) for the form designer and you can either design the form in the UI or paste in your html directly (using the Source window in the editor.

4.3.4 Validation rules

Once you have set up the data entry part of the system and started to collect data then there is time to define data quality checks that help to improve the quality of the data being collected. You can add as many validation rules as you like and these are composed of left and right side expressions that again are composed of data elements, with an operator between the two sides. Typical rules are comparing subtotals to totals of something. E.g. if you have two data elements "HIV tests taken" and "HIV test result positive" then you know that in the same form (for the same period and organisational unit) the total number of tests must always be equal or higher than the number of positive tests. These rules should be absolute rules meaning that they are mathematically correct and not just assumptions or "most of the time correct". The rules can be run in data entry, after filling each form, or as a more batch like process on multiple forms at the same time, e.g. for all facilities for the previous reporting month. The results of the tests will list all violations and the detailed values for each side of the expression where the violation occurred to make it easy to go back to data entry and correct the values.

4.3.5 Indicators

Indicators represent perhaps the most powerful data analysis feature of the DHIS2. While data elements represent the raw data (counts) being collected the indicators represent formulas providing coverage rates, incidence rates, ratios and other formula-based units of analysis. An indicator is made up of a factor (e.g. 1, 100, 100, 100 000), a numerator and a denominator, the two latter are both expressions based on one or more data elements. E.g. the indicator "BCG coverage <1 year" is defined a formula with a factor 100, a numerator ("BCG doses given to children under 1 year") and a denominator ("Target population under 1 year"). The indicator "DPT1 to DPT3 drop out rate" is a formula of 100 % x ("DPT1 doses given"- "DPT3 doses given") / ("DPT1 doses given").

Most report modules in DHIS2 support both data elements and indicators and you can also combine these in custom reports, but the important difference and strength of indicators versus raw data (data element's data values) is the ability to compare data across different geographical areas (e.g. highly populated vs rural areas) as the target population can be used in the denominator.

Indicators can be added, modified and deleted at any point in time without interfering with the data values in the database.

4.3.6 Report tables and reports

Standard reports in DHIS2 is a very flexible way of presenting the data that has been collected. Data can be aggregated by any organisational unit or orgunit level, by data element, by indicators, as well as over time (e.g. monthly, quarterly, yearly). The report tables are custom data sources for the standard reports and can be flexibly defined in the user interface and later accessed in external report designers such as iReport or BIRT. These report designs can then be set up as easily accessible one-click reports with parameters so that the users can run the same reports e.g. every month when new data is entered, and also be relevant to users at all levels as the organisational unit can be selected at the time of running the report.

4.3.7 GIS (Maps)

In the integrated GIS module you can easily display your data on maps, both on polygons (areas) and as points (health facilities), and either as data elements or indicators. By providing the coordinates of your organisational units to the system you can qucikly get up to speed with this module. See the GIS section for details on how to get started.

4.3.8 Charts and dashboard

On of the easiest way to display your indicator data is through charts. An easy to use chart dialogue will guide you through the creation of various types of charts with data on indicators, organisational units and periods of your choice. These charts can easily be added to one of the four chart sections on your dashboard and there be made easily available right after log in. Make sure to set the dashboard module as the start module in user settings.

5 Deployment Strategies

DHIS2 is a network enabled application and can be accessed over the Internet, a local intranet and as a locally installed system. The deployment alternatives for DHIS2 are in this chapter defined as i) offline deployment ii) online deployment and iii) hybrid deployment. The meaning and differences will be discussed in the following sections.

5.1 Offline Deployment

An offline deployment implies that multiple standalone offline instances are installed for end users, typically at the district level. The system is maintained primarily by the end users/district health officers who enters data and generate reports from the system running on their local server. The system will also typically be maintained by a national super-user team who pay regular visits to the district deployments. Data is moved upwards in the hierarchy by the end users producing data exchange files which are sent electronically by email or physically by mail or personal travel. (Note that the brief Internet connectivity required for sending emails does not qualify for being defined as online). This style of deployment has the obvious benefit that it works when appropriate Internet connectivity is not available. On the other side there are significant challenges with this style which are described in the following section.

- Hardware: Running stand-alone systems requires advanced hardware in terms of servers and reliable power supply to be installed, usually at district level, all over the country. This requires appropriate funding for procurement and plan for long-term maintenance.
- Software platform: Local installs implies a significant need for maintenance. From experience, the biggest challenge is viruses and other malware which tend to infect local installations in the long-run. The main reason is that end users utilize memory sticks for transporting data exchange files and documents between private computers, other workstations and the system running the application. Keeping anti-virus software and operating system patches up to date in an offline environment are challenging and bad practises in terms of security are often adopted by end users. The preferred way to overcome this issue is to run a dedicated server for the application where no memory sticks are allowed and use an Linux based operating system which is not as prone to virus infections as MS Windows.
- Software application: Being able to distribute new functionality and bug-fixes to the health information software to users are essential for maintenance and improvement of the system. Relying on the end users to perform software upgrades requires extensive training and a high level of competence on their side as upgrading software applications might be a technically challenging task. Relying on a national super-user team to maintain the software implies a lot of travelling.
- Database maintenance: A prerequisite for an efficient system is that all users enter data with a standardized meta-data set (data elements, forms etc). As with the previous point about software upgrades, distribution of changes to the meta-data set to numerous offline installations requires end user competence if the updates are sent electronically or a well-organized super-user team. Failure to keep the meta-data set synchronized will lead to loss of ability to move data from the districts and/or an inconsistent national database since the data entered for instance at the district level will not be compatible with the data at the national level.

5.2 Online deployment

An online deployment implies that a single instance of the application is set up on a server connected to the Internet. All users (clients) connect to the online central server over the Internet using a web browser. This style of deployment currently benefits from the huge investments in

and expansions of mobile networks in developing countries. This makes it possible to access online servers in even the most rural areas using mobile Internet modems (also referred to as *dongles*).

This online deployment style has huge positive implications for the implementation process and application maintenance compared to the traditional offline standalone style:

- Hardware: Hardware requirements on the end-user side are limited to a reasonably modern computer/laptop and Internet connectivity through a fixed line or a mobile modem. There is no need for a specialized server, any Internet enabled computer will be sufficient.
- Software platform: The end users only need a web browser to connect to the online server. All popular operating systems today are shipped with a web browser and there is no special requirement on what type or version. This means that if severe problems such as virus infections or software corruption occur one can always resort to re-formatting and installing the computer operating system or obtain a new computer/laptop. The user can continue with data entry where it was left and no data will be lost.
- Software application: The central server deployment style means that the application can be upgraded and maintained in a centralized fashion. When new versions of the applications are released with new features and bug-fixes it can be deployed to the single online server. All changes will then be reflected on the client side the next time end users connect over the Internet. This obviously has a huge positive impact for the process of improving the system as new features can be distributed to users immediately, all users will be accessing the same application version, and bugs and issues can be sorted out and deployed on-the-fly.
- Database maintenance: Similar to the previous point, changes to the meta-data can be done on the online server in a centralized fashion and will automatically propagate to all clients next time they connect to the server. This effectively removes the vast issues related to maintaining an upgraded and standardized meta-data set related to the traditional offline deployment style. It is extremely convenient for instance during the initial database development phase and during the annual database revision processes as end users will be accessing a consistent and standardized database even when changes occur frequently.

This approach might be problematic in cases where Internet connectivity is volatile or missing in long periods of time. DHIS2 however has certain features which requires Internet connectivity to be available only only part of the time for the system to work properly, such as the MyDatamart tool presented in a separate chapter in this guide.

5.3 Hybrid deployment

From the discussion so far one realizes that the online deployment style is favourable over the offline style but requires decent Internet connectivity where it will be used. It is important to notice that the mentioned styles can co-exist in a common deployment. It is perfectly feasible to have online as well as offline deployments within a single country. The general rule would be that districts and facilities should access the system online over the Internet where sufficient Internet connectivity exist, and offline systems should be deployed to districts where this is not the case.

Defining decent Internet connectivity precisely is hard but as a rule of thumb the download speed should be minimum 10 Kbyte/second and accessibility should be minimum 70% of the time.

In this regard mobile Internet modems which can be connected to a computer or laptop and access the mobile network are an extremely capable and feasible solution. Mobile Internet coverage is increasing rapidly all over the world, often provides excellent connectivity at low

prices and is a great alternative to to local networks and poorly maintained fixed Internet lines. Getting in contact with national mobile network companies regarding post-paid subscriptions and potential large-order benefits can be a worthwhile effort. The network coverage for each network operator in the relevant country should be investigated when deciding which deployment approach to opt for as it might differ and cover different parts of the country.

5.4 Server hosting

The online deployment approach raises the question of where and how to host the server which will run the DHIS2 application. Typically there are several options:

1. Internal hosting within the Ministry of Health
2. Hosting within a government data centre
3. Hosting through an external hosting company

The main reason for choosing the first option is often political motivation for having “physical ownership” of the database. This is perceived as important by many in order to “own” and control the data. There is also a wish to build local capacity for server administration related to sustainability of the project. This is often a donor-driven initiative as it is perceived as a concrete and helpful mission.

Regarding the second option, some places a government data centre is constructed with a view to promoting and improving the use and accessibility of public data. Another reason is that a proliferation of internal server environments is very resource demanding and it is more effective to establish centralized infrastructure and capacity.

Regarding external hosting there is lately a move towards outsourcing the operation and administration of computer resources to an external provider, where those resources are accessed over the network, popularly referred to as “cloud computing” or “software as a service”. Those resources are typically accessed over the Internet using a web browser.

The primary goal for an online server deployment is provide long-term stable and high-performance accessibility to the intended services. When deciding which option to choose for server environment there are many aspects to consider:

1. Human capacity for server administration and operation. There must be human resources with general skills in server administration and in the specific technologies used for the application providing the services. Examples of such technologies are web servers and database management platforms.
2. Reliable solutions for automated backups, including local off-server and remote backup.
3. Stable connectivity and high network bandwidth for traffic to and from the server.
4. Stable power supply including a backup solution.
5. Secure environment for the physical server regarding issues such as access, theft and fire.
6. Presence of a disaster recovery plan. This plan must contain a realistic strategy for making sure that the service will be only suffering short down-times in the events of hardware failures, network downtime and more.
7. Feasible, powerful and robust hardware.

All of these aspects must be covered in order to create an appropriate hosting environment. The hardware requirement is deliberately put last since there is a clear tendency to give it too much attention.

Looking back at the three main hosting options, experience from implementation missions in developing countries suggests that all of the hosting aspects are rarely present in option one and two at a feasible level. Reaching an acceptable level in all these aspects is challenging in terms of both human resources and money, especially when compared to the cost of option three. It has the benefit that it accommodates the mentioned political aspects and building local capacity for server administration, on the other hand can this be provided for in alternative ways.

Option three - external hosting - has the benefit that it supports all of the mentioned hosting aspects at a very affordable price. Several hosting providers - of virtual servers or software as a service - offer reliable services for running most kinds of applications. Example of such providers are Linode and Amazon Web Services. Administration of such servers happens over a network connection, which most often anyway is the case with local server administration. The physical location of the server in this case becomes irrelevant in that such providers offer services in most parts of the world. This solution is increasingly becoming the standard solution for hosting of application services. The aspect of building local capacity for server administration is compatible with this option since a local ICT team can be tasked with maintaining the externally hosted server.

An approach for combining the benefits of external hosting with the need for local hosting and physical ownership is to use an external hosting provider for the primary transactional system, while mirroring this server to a locally hosted non-critical server which is used for read-only purposes such as data analysis and accessed over the intranet.

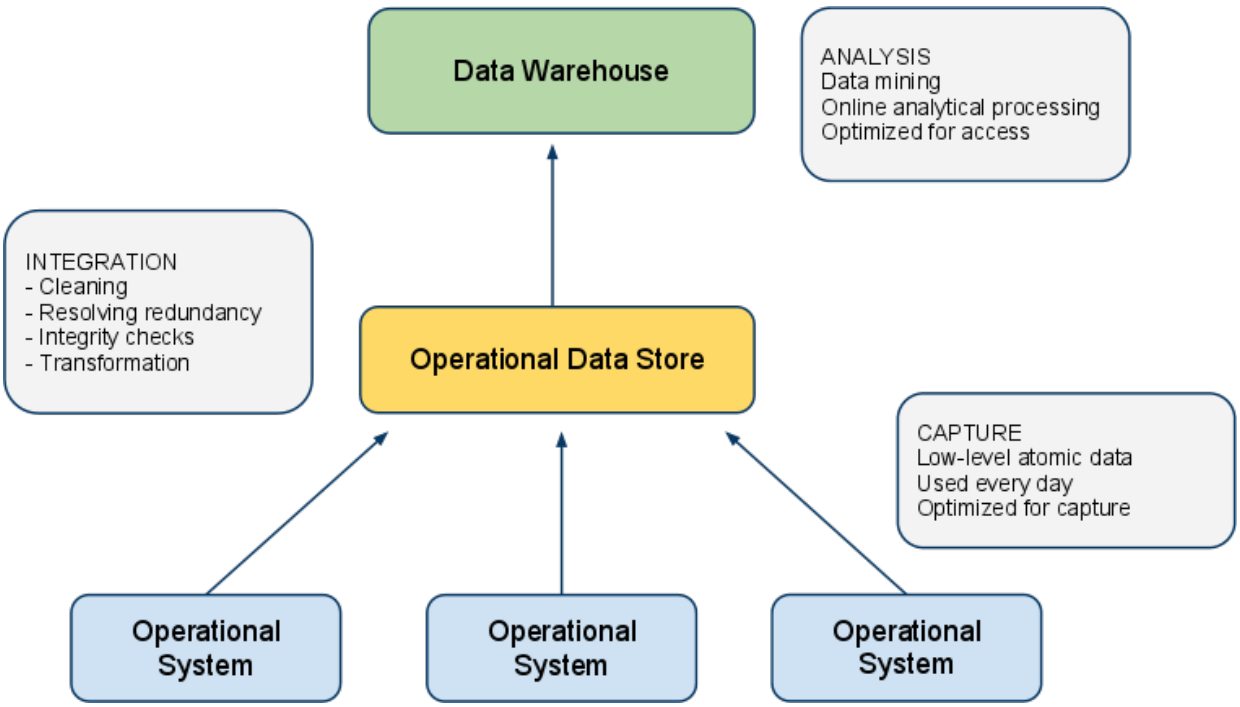
6 DHIS2 as Data Warehouse

This chapter will discuss the role and place of the DHIS2 application in a system architecture context. It will show that DHIS2 can serve the purpose of both a data warehouse and an operational system.

6.1 Data warehouses and operational systems

A *data warehouse* is commonly understood as a database used for analysis. Typically data is uploaded from various operational / transactional systems. Before data is loaded into the data warehouse it usually goes through various stages where it is cleaned for anomalies and redundancy and transformed to conform with the overall structure of the integrated database. Data is then made available for use by analysis, also known under terms such as *data mining* and *online analytical processing*. The data warehouse design is optimized for speed of data retrieval and analysis. To improve performance the data storage is often redundant in the sense that the data is stored both in its most granular form and in an aggregated (summarized) form.

A *transactional system* (or *operational system* from a data warehouse perspective) is a system that collects, stores and modifies low level data. This system is typically used on a day-to-day basis for data entry and validation. The design is optimized for fast insert and update performance.



There are several benefits of maintaining a data warehouse, some of them being:

- *Consistency:* It provides a common data model for all relevant data and acts as an abstraction over a potentially high number of data sources and feeding systems which makes it a lot easier to perform analysis.
- *Reliability:* It is detached from the sources where the data originated from and is hence not affected if data in the operational systems is purged or lost.
- *Analysis performance:* It is designed for maximum performance for data retrieval and analysis in contrast to operational system which are often optimized for data capture.

There are however also significant challenges with a data warehouse approach:

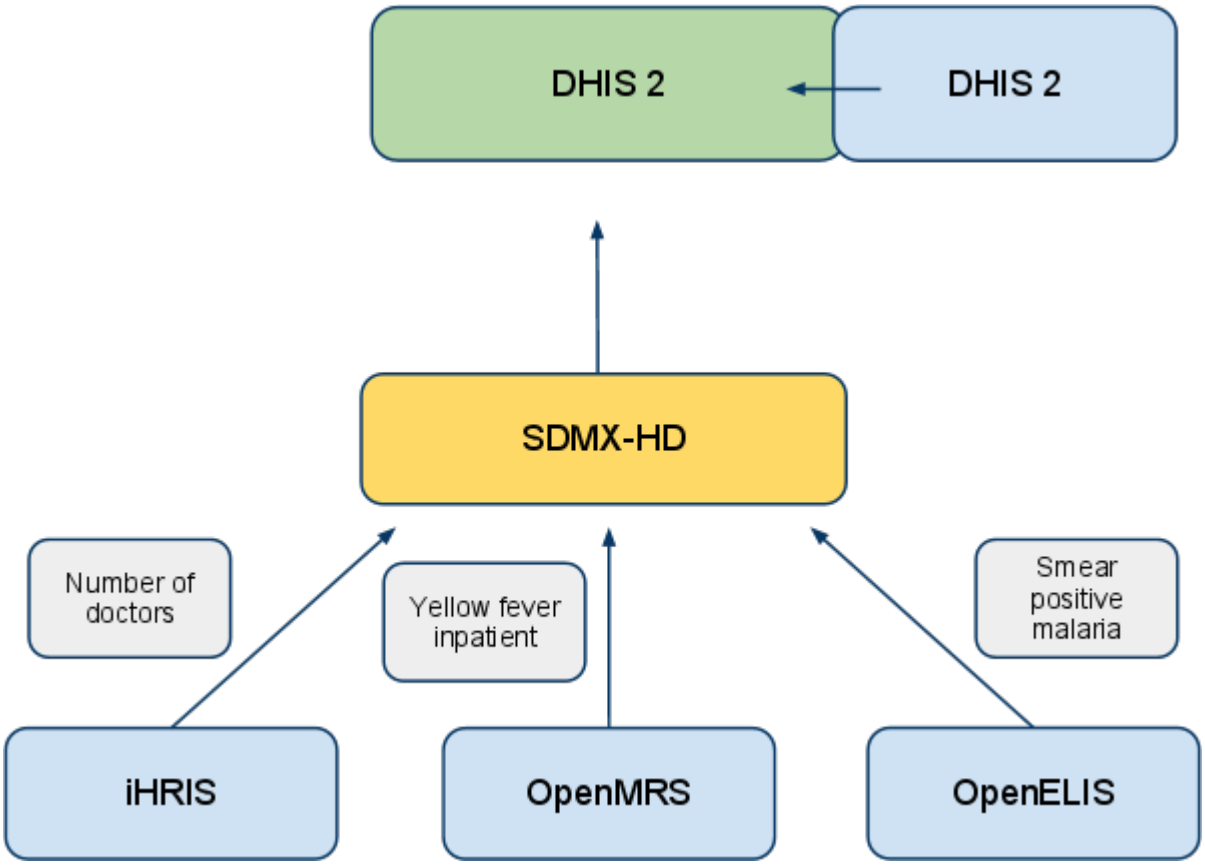
- *High cost:* There is a high cost associated with moving data from various sources into a common data warehouse, especially when the operational systems are not similar in nature. Often long-term existing systems (referred to as legacy systems) put heavy constraints on the data transformation process.
- *Data validity:* The process of moving data into the data warehouse is often complex and hence often not performed at regular and timely intervals. This will then leave the data users with out-dated and irrelevant data not suitable for planning and informed decision making.

Due to the mentioned challenges it has lately become increasingly popular to merge the functions of the data warehouse and operational system, either into a single system which performs both tasks or with tightly integrated systems hosted together. With this approach the system provides functionality for data capture and validation as well as data analysis and manages the process of converting low-level atomic data into aggregate data suitable for analysis. This sets high standards for the system and its design as it must provide appropriate performance for both of those functions; however advances in hardware and parallel processing is increasingly making such an approach feasible.

In this regard, the DHIS2 application is designed to serve as a tool for both data capture, validation, analysis and presentation of data. It provides modules for all of the mentioned aspects, including data entry functionality and a wide array of analysis tools such as reports, charts, maps, pivot tables and dashboard.

In addition, DHIS2 is a part of a suite of interoperable health information systems which covers a wide range of needs and are all open-source software. DHIS2 implements the standard for data and meta-data exhange in the health domain called SDMX-HD. There are many examples of operational systems which also implements this standard and potenitally can feed data into DHIS2:

- iHRIS: System for management of human resource data. Examples of data which is relevant for a national data warehouse captured by this system is “number of doctors”, “number of nurses” and “total number of staff”. This data is interesting to compare for instance to district performance.
- OpenMRS: Medical record system being used at hospital. This system can potentially aggregate and export data on inpatient diseases to a national data warehouse.
- OpenELIS: Laboratory enterprise information system. This system can generate and export data on number and outcome of laboratory tests.



6.2 Aggregation strategy in DHIS2

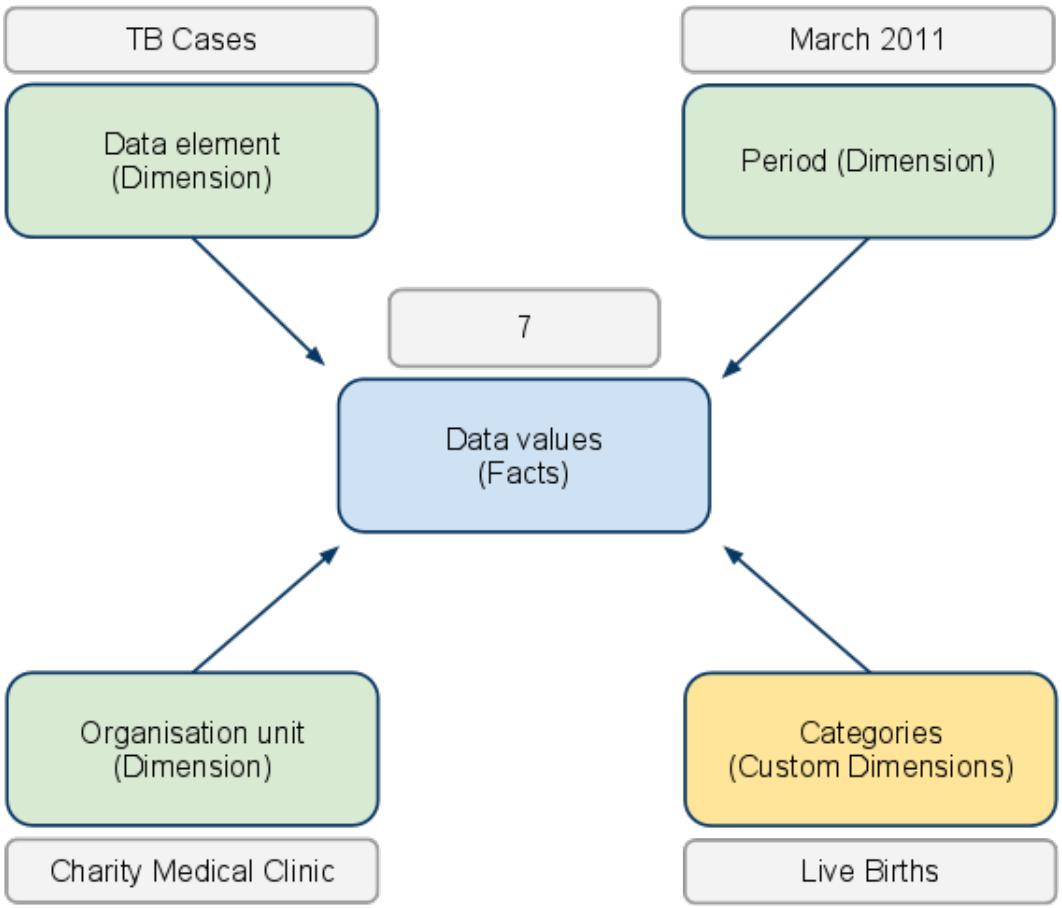
The analysis tools in DHIS2 reads aggregated data from *data mart* tables. A data mart is a data store optimized for meeting the most common user requests for data analysis. The DHIS2 data mart contains data aggregated in the*space dimension* (the organisation unit hierarchy), *time dimension* (over multiple periods) and for *indicator formulas* (mathematical expressions including data elements). Retrieving data directly from data marts provides good performance even in high-concurrency environments since most requests for analysis can be served with a single, simple database query against the data mart. The aggregation engine in DHIS2 is capable of processing low-level data in the multi-millions and managing most national-level databases, and it can be said to provide *near real-time access* to aggregate data.

DHIS2 allows for setting up scheduled aggregation tasks which typically will refresh and populate the data mart with aggregated data every night. You can choose between aggregating data for the last 12 months every night, or aggregate data for the last 6 months every night and the last 6 to 12 months every Saturday. The scheduled tasks can be configured under “Scheduling” in “Data administration” module. It is also possible to execute arbitrary data mart tasks under “Data mart” in “Reports” module.

6.3 Data storage approach

There are two leading approaches for storing data in a data warehouse, namely the *normalized* and *dimensional* approach. DHIS2 lends a bit from the former but mostly from the latter. In the dimensional approach the data is partitioned into *dimensions* and *facts*. Facts generally refers to transactional numeric data while dimensions are the reference data that gives context and meaning to the data. The strict rules of this approach makes it easy for users to understand the data warehouse structure and provides for good performance since few tables must be combined to produce meaningful analysis, while it on the other hand might make the system less flexible and harder to change.

In DHIS2 the facts corresponds to the data value object in the data model. The data value captures data as numbers, yes/no or text. The *compulsory dimensions* which give meaning to the facts are the *data element*, *organisation unit hierarchy* and *period* dimensions. These dimensions are referred to as compulsory since they must be provided for all stored data records. DHIS2 also has a custom dimensional model which makes it possible to represent any kind of dimensionality. This model must be defined prior to data capture. DHIS2 also has a flexible model of groups and group sets which makes it possible to add custom dimensionality to the compulsory dimensions after data capture has taken place. You can read more about dimensionality in DHIS2 in the chapter by the same name.



7 End-user Training

The following topics will be covered in this chapter:

- What training is needed
- Strategies for training
- Material and courses

7.1 What training is needed

In a large system like a country health information system, there will be different roles for different people. The different tasks usually depends on two factors; what the person will be doing, i.e. mainly collect data, or analyse it, or maintain the database, and where the person is located, like a facility, a district office, or at national level. A first task will then be to define the different users. The most common tasks will be:

- Data entry
- Data analysis processing, preparing reports and other information products
- Database maintenance - managing changes to the database

Data entry is typically decentralized to lower levels, such as a district. Data processing takes place at all levels, while database maintenance usually is centralized. The following table gives an example of user groups and what tasks they typically have:

Note here that many of the tasks are not directly linked to the use of DHIS2. Data analysis, data quality assessment, preparing feedback and planning regular review meetings are all integral to the functioning of the system, and should also be covered in a training strategy.

7.2 Strategies for training

To cover the wide array of tasks/users listed above, a training strategy is helpful. The majority of users will be at lower level; entering and using data. Only a few will have to know the database in-depth, usually at national level. The following are useful tips for end-user training strategies.

7.2.1 Training of trainers

Since the number of units and staff increase exponentially for each level (a country may have many provinces, each with many districts, each with many facilities), training of trainers is the first step. The number of trainers will vary, depending on the speed of implementation envisioned. As described below, both workshops and on-site training are useful, and especially for the on-site training many people will be needed.

The trainers should be at least at the level of advanced users, in addition knowing how the database is designed, how to install and troubleshoot DHIS2, and some issues of epidemiology, i.e. concepts that are useful for monitoring and evaluation of health services. As the capabilities of the staff increase, the trainers would also need to increase their skills.

7.2.2 Workshops and on-site training

Experience has showed that training both in workshops/training sessions, and on-site in real work situations are complementary. Workshops are better for training many at the same time, and are useful early on in the training sessions. Preferably the same type of users should be trained.

On-site training takes place at the work-place of the staff. It is useful to have run a more organized training session like in a workshop before, so that on-site training can focus on special issues the individual staff need more training on. Training on-site will involve less people, so it will be possible to include different types of users. An example would be a district training, where the district information officers and the district medical officer can be trained together. The communication between different users is important in the sense that it forms a common understanding of what is needed, and what is possible. Training can typically be centred around local requirements such as producing outputs (reports, charts, maps) that would be useful for local decision-support.

7.2.3 Continuation of training

Training is not a one off thing. A multi-level training strategy would aim at providing regular training as the skills of the staff increase. For example, a workshop on data entry and validation should be followed by another workshop on report generation and data analysis some time later. Regular training should also be offered to new staff, and when large changes are made to the system, such as redesign of all data collection forms.

7.3 Material and courses

There is comprehensive material available for training and courses. The main source will be the three manuals available from the DHIS2 documentation repository, to be found here.

The user documentation covers the background and purpose of DHIS2 together with instructions and explanations of how to perform data entry, system maintenance, meta-data set-up, import and export of data, aggregation, reporting and other topics related to the usage of the software. The developer documentation covers the technical architecture, the design of each module and use of the development frameworks behind DHIS2. The implementation guide is targeted at implementers and super-users and addresses subjects such as system design, database development, data harmonization, analysis, deployment, human resources needed and integration with other systems. The end user manual is a light-weight version of the user documentation meant for end users such as district records officers and data entry clerks. All can be opened/downloaded as both PDF and HTML, and are updated daily with the latest input from DHIS2 users worldwide.

The development of these guides depend on input from all users. For information how to to add content to them, please see the appendix on documentation in the User Documentation. Here you will also find information about how to make localized documentation, including versioning in different languages.

From 2011, regional workshops and courses will be held at least once a year by the international DHIS2 community. The goal is to have national technical teams working with DHIS2 customization and implementation. Sessions will also include training and capacity building by these teams in-country. End-user training, i.e. training of district M&E officers, should take place in each county by these teams.

8 Integration concepts

DHIS2 is an open platform and its implementers are active contributors to interoperability initiatives, such as openHIE. The DHIS2 application database is designed with flexibility in mind. Data structures such as data elements, organisation units, forms and user roles can be defined completely freely through the application user interface. This makes it possible for the system to be adapted to a multitude of local contexts and use-cases. DHIS2 supports many requirements for routine data capture and analysis emerging in country implementations, both for HMIS scenarios and as a basic data collection and management system in domains such as logistics, laboratory management and finance.

8.1 Integration and interoperability

Based on its platform approach, DHIS2 is able to receive and host data from different data sources and share it to other systems and reporting mechanisms. An important distinction of integration concepts is the difference between data integration and systems interoperability:

- When talking about **integration**, we think about the process of making different information systems appear as one, making electronic data available to all relevant users as well as the harmonization of definitions and dimensions so that it is possible to combine the data in useful ways.
- A related concept is **interoperability**, which is one strategy to achieve integration. We consider DHIS2 interoperable with other software applications because of its capability to exchange data. For example, DHIS2 and OpenMRS are interoperable, because they allow to share data definitions and data with each other. Interoperability depends on standards for data formats, interfaces, codes and terminologies. These would ideally be internationally agreed-upon standards, but in practice may also consist of de facto standards (which has wide acceptance and usage but is not necessarily formally balloted in a standards development organisation) and other more local agreements within a particular context.

DHIS2 is often used as an integrated data warehouse, since it contains (aggregate) data from various sources, such as Mother and Child health, Malaria program, census data, and data on stocks and human resources. These data sources share the same platform, DHIS2, and are available all from the same place. These subsystems are thus considered integrated into one system.

Interoperability in addition will integrate data sources from other software applications. For example, if census data is stored in a specialized civil registry or in a vital events system, interoperability between this database and DHIS2 would mean that census data would also be accessible in DHIS2.

Finally, the most basic integration activity (that is not always taken into account in the interoperability discussion) is the possibility to integratedata from existing paper systems or parallel vertical systems into DHIS2. Data will be entered directly into DHIS2 without passing through a different software application. This process is based on creating consistent indicator definitions and can already greatly reduce fragmentation and enhance data analysis through an integrated data repository.

8.2 Objectives of integration

In most countries we find many different, **isolated** health information systems, causing many information management challenges. Public Health Information System have seen an explosive and often uncoordinated growth over the last years. Modern information technology makes it less costly to implement ICT4D solutions, which can lead to a high diversity of solutions. A staggering example was the mHealth moratorium declaration of Uganda´s MoH in 2012, as a

reaction to an avalanche of around 50 mHealth solutions that were implemented within the course of a few years. Most of these solutions were standalone approaches that did not share their data with the national systems and rarely were developed beyond pilot status.

This may lead to the conclusion, that all systems should be connected or that **interoperability is an objective** in itself. However DHIS2 is often employed in contexts, where infrastructure is weak, and where resources to run even basic systems reliably are scarce. Fragmentation is a serious problem in this context, however interoperability approaches can only resolve some of the fragmentation problems - and often interoperability approaches result in an additional layer of complexity.

Example: Complexity of Logistics solutions in Ghana

In the area of Logistics or Supply Chain Management, often a multitude of parallel, overlapping or competing software solutions can be found in a single country. As identified in a JSI study in 2012, eighteen (18!) different software tools were documented as being used within the public health supply chain in Ghana alone.

Systems interoperability therefore seems as one possibility to remove fragmentation and redundancies and give public health officers a concise and balanced picture from available data sources. However the effort of connecting many redundant software solutions would be very high and therefore seems questionable. In a first step, focus should be on **reducing the number of parallel systems**and identifying the most relevant systems, afterwards these relevant systems can be integrated.

On this background, we want to define the major objectives of DHIS2 integration approaches:

- **Calculation of indicators**: Many indicators are based on numerators and denominators from different data sources. Examples include mortality rates, including some mortality data as numerator and population data as denominator, staff coverage and staff workload rates (human resource data, and population and headcount data), immunization rates, and the like. For the calculation, you need both the numerator and denominator data, and they should thus be integrated into a single data warehouse. The more data sources that are integrated, the more indicators can be generated from the central repository.
- **Reduce manual processing** and entering of data: With different data at the same place, there is no need to manually extract and process indicators, or re-enter data into the data warehouse. Especially interoperability between systems of different data types (such as patient registers and aggregate data warehouse) allows software for subsystems to both calculate and share data electronically. This reduces the amount of manual steps involved in data processing, which increases data quality.
- **Reduce redundancies**: Often overlapping and redundant data is being captured by the various parallel systems. For instance will HIV/AIDS related data elements be captured both by both multiple general counselling and testing programs and the specialized HIV/AIDS program. Harmonizing the data collection tools of such programs will reduce the total workload of the end users. This implies that such data sources can be integrated into DHIS2 and harmonized with the existing data elements, which involves both data entry and data analysis requirements.
- Improve **organizational aspects**: If all data can be handled by one unit in the ministry of health, instead of various subsystems maintained by the several health programs, this one unit can be professionalized. With staff which sole responsibility is data management,

processing, and analysis, more specialized skills can be developed and the information handling be rationalized.

- Integration of **vertical programs**: The typical government health domain has a lot of existing players and systems. An integrated database containing data from various sources becomes more valuable and useful than fragmented and isolated ones. For instance when analysis of epidemiological data is combined with specialized HIV/AIDS, TB, financial and human resource data, or when immunization is combined with logistics/stock data, it will give a more complete picture of the situation.

DHIS2 can help streamlining and **simplifying system architecture**, following questions such as:What is the objective of the integration effort? Can DHIS2 help reduce the number of systems? Can an DHIS2 integration help provide relevant management information at a lower cost, at at higher speed and with a better data quality than the existing systems? Is DHIS2 the best tool to replace other systems, or is another fit-for-purpose solution that can interoperate with DHIS2 more appropriate? More practical information on defining these objectives can be found in *STEP 1 of the 6-Step implementation guideline*.

8.3 Health information exchange

Since there are different use-cases for health information, there are different types of software applications functioning within the health sector. We use the term architecture for health information to describe a plan or overview of the various software applications, their specific uses and data connections. The architecture functions as a plan to coordinate the development and interoperability of various subsystems within the larger health information system. It is advisable to develop a plan that covers all components, including the areas that are currently not running any software, to be able to adequately see the requirements in terms of data sharing. These requirements should then be part of specifications for the software once it is developed or procured.

The **openhealth information exchange (openHIE)** is an interoperable interpretation of this architecture, with an HMIS or DHIS2 often assuming a significant role in it.The openHIE framework has been developed with a clear focus on countries in low resource settings, through the participation of several institutions and development partners, including the Oslo HISP program.

The schematic overview below shows the main elements of the openHIE framework, containing a component layer, an interoperability services layer and external systems. The openHIE component layer covers meta or reference data (Terminology, Clients, Facilities), Personal data (Staff, Patient History) and national health statistics. The purpose is to ensure the availability of the same meta data in all systems that participate in the corresponding data exchange (e.g. indicator definitions, facility naming, coding and classification). In some cases, like the case of the Master Facility Registry, the data may also be used to provide information to the general public through a web portal. While the interoperability layer ensures data brokerage between the different systems, the external systems layer contains several sub-systems, many at point of service level, with often overlapping functional range.

There are different approaches to define an eHealth architecture. In the context of this DHIS2 guideline, we distinguish between approaches based on a 1:1 connection versus approaches based on an n:n connection (many-to-many).

8.3.1 *1:1* integration

In many countries a national HMIS is often the first system to be rolled out to a large number of facilities and to manage a large number of data on a monthly or quarterly basis. When countries start to develop their health system architecture further, DHIS2 often will be connected to some

other systems. This connection is often done directly through a simple script, which automates a data transfer.

We talk of a 1:1 connection because it is limited to two systems. In the case of an LMIS/HMIS integration, one LMIS will transfer data to DHIS2 as defined in the script. This hands-on approach often represents a first step and is one of the most common use cases on the way to an interoperable openHIE architecture. However, this simplicity also brings along disadvantages: In case a second logistics system would want to transfer data to DHIS2 (e.g. commodity data for a specific disease program), a second script would have to be written, to perform this task. These two scripts would then run independently from another, resulting in two separate 1:1 connections.

8.3.2 *n:n* integration

A different approach is based on placing a purpose-built software to serve as an **interoperability layer** or BUS approach, managing the data transfer between possibly several systems on either side (n:n). This could be the case if for example you wanted to collect stock level data through different LMIS applications, and then share it to a central warehouse LMIS, the HMIS and some vertical disease programs system. The openHIE reference software to assume this role is Jembi , but systems like "MOTECH" have also been used for this purpose as will be discussed below.

While this approach may result in a higher initial effort, it promises to make further integration project easier, because the interoperability layer is being alimented with definitions and mappings that can be re-used for connecting the next systems.

In practice, there are certain challenges to this approach. It takes a considerable effort of qualified resources to activate APIs and with each new release of any involved system, data flows require re-testing and if necessary adaptations. Also, to be successful these implementation projects typically have to go through a series of *complex steps*, such as the agreement on an interoperability approach embedded in the national eHealth strategy, the definition of data standards and sustainable maintenance structure, and attaining a stakeholder consensus on data ownership and sharing policies. There can be some long term consequences when data and systems are knitted together - it creates new roles, jobs and tasks which didn't exist before and may not have been planned for (metadata governance, complex system administration, boundary negotiators, etc.).

Example: Grameen DHIS2/CommCare middle layer in Senegal

In a Integration concepts, MOTECH serves as technical middle layer between an LMIS for mobile data collection at the health facility level (CommCare) and DHIS2, allowing to define data mapping, transformation rules and data quality checks. The interface is set-up to transfers data from CommCare Supply to DHIS2 whenever data is saved into a CommCare form at facilities. For each commodity, data on consumption, available stock, losses and stock-out data is transferred from CommCare to DHIS2.

The higher initial investment of the Senegal approach hints towards a more ambitious long-term system architecture, foreseeing that the MOTECH platform may in future serve to accommodate further interoperability task. However we do not see any of the country activities tightly embedded in a text-book eHealth architecture, which would clearly define areas of priority, leading systems for each priority and the relations and resulting APIs between these different components. One may argue that interoperability projects are built on a weak foundation if there is no previous consensus on an architectural master plan. On the other hand it is also valuable to allow system initiatives to organically develop, as long as they are rooted in well-founded country needs.

8.3.3 Architecture, standards and mapping

An important element of an eHealth architecture is the inclusion of **international eHealth standards**. Standards are especially relevant for n:n connections, less so for direct (1:1) connections.

Some standards are on the technical level (e.g. transmission methods), other on the contents side (e.g. WHO 100 core indicators). Gradually aligning national system initiatives to these standards can give countries access to proven solutions, benefitting from medical and technological innovation.

> **Example: Ghana EPI**
>
> The Ghana case illustrates how the WHO EPI reporting requirements serves to define standard data in DHIS2. This standardization at the dataset and terminological level is the basis for the system integration. In the area of DHIS2, work is ongoing with WHO to develop standardized datasets, which could in the future open up new opportunities for interoperability and efficiency gains by offering some consistency of metadata across systems, and also encouraging countries to reuse existing solutions.

At the **language** level, there is a need to be consistent about definitions. If you have two data sources for the same data, they need to be comparable. For example, if you collect malaria data from both standard clinics and from hospitals, this data needs to describe the same thing if they need to be combined for totals and indicators. If a hospital is reporting malaria cases by sex but not age group, and other clinics are reporting by age group but not sex, this data cannot be analysed according to either of these dimensions (even though a total amount of cases can be calculated). There is thus a need to agree on uniform definitions.

In addition to uniform definitions across the various sub-systems, **data exchange standards** must be adopted if data is to be shared electronically. The various software applications would need this to be able to understand each other. DHIS2 is supporting several data formats for import and export, including the most relevant standard ADX. Other software applications are also supporting this, and it allows the sharing of data definitions and aggregate data between them. For DHIS2, this means it supports import of aggregate data that are supplied by other applications, such as OpenMRS (for patient management) and iHRIS (for human resources management).

A crucial element of the architecture is how organize data **mapping**. Typically the metadata of different systems does not match exactly. Unless an MoH has been enforcing a consequent data standard policy, different systems will have different codes and labels for a facility. one System may call it *District Hospital - 123*, the other system may refer to it as Malaria Treatment Centre - 15. To be able to transfer data, somewhere the information that these two facilities correspond needs to be stored.

In the case of a 1:1 connection, this mapping has to be done and maintained for every connection, in case of an n:n interoperability approach, one side of the definitions can be re-used.

In order to assure that the data can flow smoothly, you need to have clear responsibilities on both sides of the system regarding data maintenance and troubleshooting. For example, there need to be previously defined standard procedures for such activities as adding, renaming, temporarily deactivating or removing a facility on either of the two systems. Changes of database

fields that are included in a transferred data record need also to be coordinated in a systematic way.

8.4 Aggregate and transactional data

DHIS2 has been expanding its reach into many health systems. Starting from its familiar grounds of aggregate data sets for routine data it has included patient related data and then data in the areas of HR, finance, logistics and laboratory management, moving towards operational or transactional data.

We can differentiate between transactional and aggregate data. A **transactional system** (or operational system from a data warehouse perspective) is a system that collects, stores and modifies detailed level data. This system is typically used on a day-to-day basis for data entry and validation. The design is optimized for fast insert and update performance. DHIS2 can incorporate aggregate data from external data sources, typically aggregated in the space dimension (the organisation unit hierarchy), time dimension (over multiple periods) and for indicator formulas (mathematical expressions including data elements).

When we look at a transactional system, such as a logistics software for the entire supply chain or parts of it, there is one fundamental decision to take: Do you need to track all detailed transactions at all levels, including such operations as returns, transfer between facilities, barcode reading, batch and expiry management? Or can you get most of your needed decision insight results using aggregate data?

Supply chains may often be well monitored and to some degree, managed, as long as data are reliably available where and when they are needed for operational decisions and for monitoring purposes.. The main indicators *intake, consumption and stock level at the end of period*can be managed without electronic transactions and often suffice to give the big picture of system performance, and may reduce the needs for system investment.

Being realistic about what data need to be collected, how often, and who will be using them is important so you don't create systems that fail due to lack of use or unrealistic expectations about how the data will be used. Digital logistics management systems can work well when they are fully integrated into routine workflows and designed to make the users' jobs easier or more efficient.

> **Note**
>
> The expectation, that more detailed data leads to better logistics management is not always fulfilled. Sometimes the ambitious attempt to regularly collect logistics transaction data results in less data quality, for example because the data recording, which may have to happen on a daily basis instead of a monthly or quarterly basis, is not carried out reliably. On the other hand, if the transactional system is well maintained and monitored, more detailed data can help identify inaccuracies and data quality challenges, reduce wastage (due to expiry or CCE failure), support a recall, manage performance and lead to improvements in supply chain decision making. Analysing detailed data may help to discover root causes of some problems and improve the data quality in the long run.

DHIS2 can assume different roles in interoperability scenarios. A common interoperability scenario is for DHIS2 to receive aggregate data from an operational system, in which case the operational system adds up the transactions before passing it on to DHIS2. However, DHIS2 may to a certain extent also be configured to store detailed transactional data, receiving it from external systems or through direct data entry in DHIS2.

On this basis we try making a comparative overview, comparing aggregate DHIS2 data management with data management of external specialized system. This can serve as a rough orientation, but is not static since both the capabilities of DHIS2 and its interpretation by implementers are broadening with almost each release.

Area	Aggregate DHIS2	External specialized systems
Logistics	Aggregate data, e.g. end-of-month facility stock levels can be send through DHIS2. DHIS2 can produce simple stock level and consumption reports.	Supply chain management support logistics system operations and can track detailed stock movements (Issuing, resupplying, allocating, wastage) and record details such as production batch numbers. SCM systems create forecasting, replenishment and elaborate control reports, allowing for real time monitoring of stock levels, notifications (low stock, workflow management, CCE failure, etc.), supported estimations, and emergency orders.
Finance	Aggregate data, e.g. on total expenditure or cash level can be send through DHIS2. DHIS2 can produce simple finance overview reports, e.g. on remaining budgets.	Finance management systems allow fully traceable recording of financial transactions according to legal requirements, including budgeting, transfers, cancellations, reimbursements etc. Multi-dimensional tagging of transactions allows for analytical reports.
Patient tracking	Disease or program related data are collected by DHIS2, DHIS2 Tracker also allows a simplified longitudinal view on medical records, including patient history and multi-stage clinical pathways.	Specialized hospital management systems can cover and optimize complex workflows between different departments (e.g. reception, payment counter, wards, OPD, IPD, laboratory, imaging, storeroom, finance and HR administration, medical device maintenance, etc.).
Human Resources	DHIS2 collects human resource related indicators, for example planned positions and vacancies per facility.	A specialized HR management system can track detailed status information and changes for a Health Worker (accreditation, promotion, sabbatical, change of position, change of location, additional training, etc.). It comes with pre-designed reports for both operational oversight and planning.

8.5 Different DHIS2 integration scenarios

The different objectives described above lead to different integration scenarios. DHIS2 can assume multiple **roles** in a system architecture:

- Data input: data entry (offline, mobile), data import (transactional data, aggregate data)
- Data storage, visualisation and analysis with in-built tools (DWH, reports, GIS)
- Data sharing to external tools (e.g. DVDMT), via web APIs, web apps

In the following paragraphs we discuss the data input and data sharing approaches, then we present the example of the vertical integration where DHIS2 often assumes all these roles.

The role of DHIS2 to store, visualise and analyse data is discussed seperpately in the data warehouse section.

8.5.1 Data input

There are several aspects on how DHIS2 deals with data input. On the most basic level, DHIS2 serves to replace or at least mirror paper-based data collection forms, integrating the data electronically. This will result in **manual data entry** activities at facility or at health administration level. The next input option is to **import data**. DHIS2 allows to import data through a user interface, which is a method requiring little technical knowledge, but needs to be executed manually every time data needs to be made available. A detailed description of the import functions can be found in the DHIS2 user guides.

Practical note:

The manual data entry and import approach require relatively little technical effort. They may also be used temporarily to pilot a data integration approach. This allows user to test indicators and reports, without having to employ dedicated technical resources for the development of automated interoperability functions, either through a 1:1 or an n:n connection.

8.5.2 Data sharing

There are three sharing scenarios, (1) a simple *data export*, (2) *DHIS2 apps and (3) external apps or websites connecting to the DHIS Web API*. Similar to the import functions described in the data input section, the most accessible way of data sharing is to use the data export functions that are available from the user menu, requiring little technical knowledge.

Due to its modular design DHIS2 can be extended with **additional software modules, which can be downloaded from the DHIS2***App store*. These software modules can live side by side with the core modules of DHIS2 and can be integrated into the DHIS2 portal and menu system. This is a powerful feature as it makes it possible to extend the system with extra functionality when needed, typically for country specific requirements as earlier pointed out.

The downside of the software module extensibility is that it puts several constraints on the development process. The developers creating the extra functionality are limited to the DHIS2 technology in terms of programming language and software frameworks, in addition to the constraints put on the design of modules by the DHIS2 portal solution. Also, these modules must be included in the DHIS2 software when the software is built and deployed on the web server, not dynamically during run-time.

In order to overcome these limitations and achieve a looser coupling between the DHIS2 service layer and additional software artefacts, the DHIS2 development team decided to create a **Web API**. This Web API complies with the rules of the REST architectural style. This implies that:

- The Web API provides a navigable and machine-readable interface to the complete DHIS2 data model. For instance, one can access the full list of data elements, then navigate using the provided hyperlink to a particular data element of interest, then navigate using the provided hyperlink to the list of forms which this data element is part of. E.g. clients will only do state transitions using the hyperlinks which are dynamically embedded in the responses.
- Data is accessed through a uniform interface (URLs) using a well-known protocol. There are no fancy transport formats or protocols involved - just the well-tested, well-understood HTTP protocol which is the main building block of the Web today. This implies that third-party developers can develop software using the DHIS2 data model and data without knowing the DHIS2 specific technology or complying with the DHIS2 design constraints.
- All data including meta-data, reports, maps and charts, known as resources in REST terminology, can be retrieved in most of the popular representation formats of the Web of today, such as HTML, XML, JSON, PDF and PNG. These formats are widely supported in applications and programming languages and gives third-party developers a wide range of implementation options.

This Web API can be accessed by different external information system. The effort needed for developing new information systems and maintaining them over time is often largely underestimated. Instead of starting from scratch, a new application can be built on top of the Web API.

Extenal systems can offer different options for visualizing and presenting DHIS2 data, e.g. in the form of dashboards, GIS and charting components. Web portals targeted at the health domain can use DHIS2 as the main source for aggregate data. The portal can connect to the Web API and communicate with relevant resources such as maps, charts, reports, tables and static documents. These data views can dynamically visualize aggregate data based on queries on the organisation unit, indicator or period dimension. The portal can add value to the information accessibility in several ways. It can be structured in a user-friendly way and make data accessible to inexperienced users. An example for this is the Tanzania HMIS Web Portal.

8.6 DHIS2 maturity model

Taking into account all the above elements on system architecture and data types, DHIS2 implementers have several options on how to approach implementations:

- Focus on transactional or aggregate data
- Focus on data integration or systems interoperability

Given the efforts required to implement systems interoperability, many Ministries of Health are going for the pragmatic shortcut of integrating data such as basic stock level data **directly into their existing national DHIS2**. As a rapidly evolving platform, DHIS2 has been adding a lot of functionality over the last years, especially in DHIS2 Tracker. Taking the example of logistics data, the following main functions are currently available:

- Data entry form mirroring the widely used Report and Requisition (R&R) paper form. Data entry by facilities is possible through the desktop browser or a mobile app, including in offline mode. These electronic forms can be filled by staff based on the paper stock cards, that are normally placed next to the commodity in the store room.

- DHIS2 can then produce reports for central level performance monitoring, giving commodity and program managers an understanding of how the logistics system is functioning.. Depending on how the logistics system operates, these data may also be able to support operational decision-making although a more complete analysis of logistics business processes and users should be conducted first.
- Stock data can be transformed into logistics indicators, that can be put into context with other program indicators, for example cross-referencing number of patients treated with a specific pathology and corresponding drug consumption.

Although each country that we look at in the use cases has their own development path towards system integration, some common learnings can be drawn from their experiences. The maturity model below describes an evolutionary approach to cope with integration and interoperability challenges, allowing the different stakeholders in a national Health System to grow professional analytics and data usage habits.

The maturity model suggests moving from aggregate data to transactional data and from stand-alone to interoperable systems (using the example of logistics data).

1. DHIS2 is often one of the first systems to cover the health administration and several facility levels of a country. At first core disease indicators are covered (for example corresponding to the 100 WHO Core Health Indicators).
2. In a second phase, different stakeholders seek to complement the disease and service delivery data they are reporting with basic LMIS data. This can be done on an aggregate basis in DHIS2, e.g. by including stock levels and consumption in periodic reports. This will provide high level information on logistics system performance but may or may not provide sufficient insights to support improved logistics system operations.
3. At a more mature stage, there may be a legitimate need for specialized logistics systems, especially when a very detailed transactional view is wanted to have a more granular control, (e.g. returns, transfers between facilities, batch numbers and expiries, etc.). DHIS2 Tracker can offer some event or patient related data management functions, but cannot always achieve the degree of workflow support provided by other, more specialized solutions.
4. In a mature technological and managerial environment, the logistics transactions can be shared to DHIS2 in an aggregate form, moving from a stand-alone to an integrated scenario.

8.7 Implementation steps for successful data and system integration

The purpose of this step-by-step DHIS2 Implementation Guide is to provide a methodology for implementers to create and support a DHIS2 integration scenario. The guide is based on the best practices and lessons learned. The guide advocates for a country driven, iterative, and agile approach that begins with collecting user stories and functional requirements. The guide is intended as a framework that can be adapted to the specific context of each country. The content describes specific examples for each step detailing user stories, data specifications, job aids and checklists to guide the use of the reference implementation software. The basic structure, including the 6 steps, are basedon the OpenHIE implementation methodology:

Step 1: Identify Stakeholders and Motivations for Improved Facility Data

Step 2: Document Facility Registry Specifications and User Stories

Step 3: Set Up Initial Instance

Step 4: Identify Gaps & Iterative Development via User Testing

Step 5: Scaling the Registry Implementation

Step 6: Provide Ongoing Support

In addition to these steps related to interoperability, it is also interesting to reference back to some of the general DHIS2 implementation experiences and best practises given in the sections on Recommendations for National HIS Implementations and Setting Up a New Database. A typical DHIS2 implementation approach which is also vital for interoperability projects is a **participatory** approach. This approach stresses to include right from project start a local team with different skills and background to assume responsibility as soon as possible.

8.7.1 Step 1: Define strategy, stakeholders and data usage objectives

In a first step, the objectives of the integration project will be defined. As with every technology project, there should be a clear **consensus on strategic and functional objectives**. Technological innovation and feasibility should not be the sole driving force but rather a clearly defined organisational goal. Therefore this step is also intended to answer the question: "Why do we want to connect systems or integrate data from different sources with DHIS2?"

On a practical level, this leads to questions on the data integration approach, such as:

- Do you want to eliminate paper forms or even eliminate data sets that are redundant or not needed anymore?
- Can you integrate the (aggregate) data into DHIS2?
- Can you integrate the detailed (e.g. patient level or transactional) data into DHIS2, using DHIS2 tracker functions?
- If you want to create a data exchange connection between DHIS2 and another system, how do you define ownership and responsibilities?

Activities to answer this question are described below and will lay the groundwork for an DHIS2 interoperability project.

8.7.1.1 Identify Stakeholders and Motivations

It is in the nature of interoperability projects to have more than one stakeholder. Stakeholders from different areas need to agree on a common system approach, for example the team responsible for the national HMIS (e.g. the M&E department or Planning Department) and the Logistics Department in case of an LMIS implementation. These two main areas often have sub-divisions, e.g. in the logistics area the procurement unit, the warehousing unit, the transport unit. In addition, stakeholders from disease specific programs will have their own regimens and commodity managers. In addition to these local actors, international partners (agencies, donors, iNGOs, consultancies) are often also involved in the decision making process.

Therefore it´s interesting to look at the main motivations of the stakeholders and how to mitigate risks resulting from potential diverging interests.

- Central MoH Departments such as **M&E&**Planning often are the main stakeholders for a standardisation of indicators and IT Systems
- **Central IT departments** have a general interest over (often locally controlled) technology choices and ownership, hardware and software purchases. They are often dealing with network and hardware issues but lack experience dealing with complex web-based architectures and data exchanges.

- **Specialized disease programs** are often under pressure to deliver very program specific indicators, both for their own management but also responding to donor driven approaches. They may also feel more comfortable controlling their proper IT system to be sure their needs are prioritized.
- **Specialized functional areas** (such as Human Resources, Logistics, Hospital Management) are often in a sandwich position, having to cater to the information needs of several different stakeholders, while trying to achieve operational efficiency with limited resources.

By identifying who is interested to provide or utilize the data, the lead implementers can start to form a project team to inform the design and implementation. One method for characterizing stakeholders involves grouping interested parties by their functional roles. The existing infrastructure and procedures are also important to understanding governance and curation options. Understanding the stakeholders and their corresponding systems is a critical first step.

8.7.1.2 eHealth System inventory

It is important to get a clear view on the overall IT systems landscape. This can help make sure that interoperability investment is done to strengthen the main systems and that the investments contribute to a **simplification** of the system architecture. For example, if the system inventory shows that there are a lot of redundant functional systems, e.g. more than 10 different logistics systems or modules in a country, the interoperability project should try to contribute to a mid or long-term rationalization of this situation. This could mean to participate in a national consensus finding process to identify the most future-proof solutions, identify national "champions" for each speciality and develop a roadmap for aligning these systems or data and removing underutilized or redundant systems.

Also in this context it is interesting to analyse whether simple indicators can be collected and managed in DHIS2 itself and how this can complement logistics system improvement efforts (as this is later explained in an LMIS example). Once the stable and sustainable systems have been identified, planning for a data exchange with DHIS2 can start.

8.7.1.3 Explore Opportunities and Challenges

The motivations driving an implementation can be detailed by the perceived opportunities or challenges that stakeholders face. This might include the desire to share data across systems related to health facilities for supply chain management, monitoring and evaluation, health service delivery and many other systems. User stories and use cases will be documented in depth during Step 2, but a high level vision of motivations to engage with partners is also needed.

8.7.1.4 Organisation and HR

Clear national policies on data integration, data ownership, routines for data collection, processing, and sharing, should be in place at the start of the project. Often some period of disturbance to the normal data flow will take place during integration, so for many the long-term prospects of a more efficient system will have to be judged against the short-term disturbance. Integration is thus often a stepwise process, where measures need to be taken for this to happen as smoothly as possible.

Country example: Ghana CHIM

- **Stakeholder cooperation**: The Ghana *Centre for Health Information Management*(CHIM) has a clear position towards vertical programs and other partners with proper software initiatives. CHIM establishes DHIS2 as an attractive data collection option, supporting other GHS stakeholders to connect to DHIS2 and to work on a common interoperability strategy, evolving DHIS2 according to stakeholder needs. **This also includes data sharing agreements**.
- **Strong sense of system ownership**: CHIM has a strong determination to build up the necessary know-how inside the CHIM team to configure and maintain the system. The CHIM team consists of Health Information Officers, that combine Public Health and Data Management skills.

Also, having clearly defined **system maintenance and update procedures** can certainly help to manage interoperability.

Country example: Ghana CHIM

As an example, in the case of Ghana DHIS2, a clear yearly system update cycle is in place: Towards the end of each year, new indicators are created and the corresponding paper forms are issued. Staff will receive training and is prepared for data entry. The new form for EPI data was included in this update cycle and EPI staff was prepared for data entry as part of the process. This systematic procedure allows GHS to quickly respond to the needs of stakeholders such as the EPI Programme and accommodate their data and reporting needs with a limited and predictable investment. It puts CHIM in a position to contribute to the rationalization and simplification of the national Health System Architecture, gradually integrating the data management for more **vertical programs**, both on the side of data entry and analytics.

A key principle for HISP is to engage the local team in building the system from the very beginning, with guidance from external experts if needed, and not to delay knowledge transfer towards the end of the implementation. Ownership comes first of all from building the system and owning every step of this process.

8.7.2 Step 2: Document Specifications and Requirements

- Collect existing metadata
- Document data specifications
- Document user stories

8.7.3 Step 3: Carry Out Specifications and Identify Gaps

- Implement the specifications
- Identify and piroritize incomplete user stories

8.7.4 Step 4: Iteration and User Testing

- Agile and iterative development
- User testing

- Collect, reconcile and upload data

8.7.5 Step 5: Scale-Up

- Confirm user roles and responsibilities
- User training
- Critical integrations

8.7.6 Step 6: Ongoing Support

While during the implementation phase a temporary support structure should be available, afterwards a permanent support structure needs to be set-up. The main challenge is to have clear responsibilities. In an ideal situation, we are dealing with two stable systems that each have already their own clearly defined support structure.

However in reality some recurring challenges may have to be dealt with: Many Public Health System are undergoing dynamic developments, leading to changes in data collection needs or calculation of indicators.

Interoperability tends to be a tedious technical and organisational charge. All of the three described initiatives have consumed a considerable effort of qualified **resources** to activate APIs. In addition, with each new release of any involved system, data flows require re-testing and if necessary adaptations. To be successful these implementation projects typically have to go through a series of complex steps, such as the agreement on an interoperability approach embedded in the national eHealth strategy, the definition of data standards and sustainable maintenance structure, and attaining a stakeholder consensus on data ownership and sharing policies. There can be some long term consequences when data and systems are knitted together - it creates **new roles, tasks and categories of labour** which need to be planned for (metadata governance, complex system administration, boundary negotiators, etc.). A solution could be to discuss the new responsibilities beforehand, assigning them to job descriptions, teams and specific positions.

8.7.6.1 Metadata responsibility

Another important area is that of **metadata governance**, particularly in the scenarios of secondary use of data. In a stand-alone set-up, metadata, such as facility or commodity codes can be managed without much consideration of other stakeholder´s needs. But in an interoperability environment, metadata changes will have effects outside of the individual system. Metadata governance can be highly formalised through registries or more manual through human processes.

In order to determine the appropriate approach, is it useful to estimate the expected *metadata maintenance effort* and the consequences of unsynchronized metadata across different systems. In the case of the LMIS/DHIS2 integrations, there are potentially thousands of facility identifiers that could go out of synch. However normally, facility identifiers do not change often since the physical infrastructure of most public health system is relatively constant. As to the commodities, although regimes and priority drugs may change over time, the number of datasets is relatively small: The commodity list of a program often contains less than 20 products. Therefore often it can be practical to update a commodity manually, and not invest into an interoperability solutions such as an automated metadata synchronization.

8.8 Specific integration and interoperability use cases

DHIS2 has been expanding its reach into many health systems. Starting from its familiar grounds of aggregate data sets for routine data it has included patient related data and then data in the

areas of HR, finance, logistics and laboratory management. This is in line with the development of DHIS2 in many country settings, where implementers are pushing the use beyond its originally intended scope.

This is also reflected in the overall system architecture. Since the expanding functionality of DHIS2 reduces the urgency to introduce or maintain other specialized systems, the number of potential data interfaces decreases. This **reduced complexity** in system architecture is certainly a benefit for a Health System with limited resources.

For several years now, DHIS2 has grown its data management activities organically, allowing the actual usage to lead to sometimes unforeseen solutions. However, there are also limits to where leveraging DHIS2 seems useful. In the following sections, special systems will be described.

8.8.1 Logistics Management

a) Introduction

Logistics Management Systems **(LMIS)** or Supply Chain Management Systems **(SCM)** serve to replace paper systems to increase standardization, transparency, timeliness of procurement, efficiency, safety, cost-effectiveness, and to reduce waste. National SCMS/LMIS can cover such functions as commodity planning, budgeting, procurement, storage, distribution and replenishment of essential drugs and consumables.

b) Implementing LMIS in DHIS2

Supply chains can often be well controlled with aggregate data only, as long as data is provided reliably from all relevant levels and followed up upon. The main indicators intake, consumption and stock level at the end of period can be managed without electronic transactions and often suffice to give the big picture, reducing the needs for system investment. As a rapidly evolving platform, DHIS2 has been adding a lot of functionality over the last years, especially in DHIS2 Tracker. The following main functions are currently available:

- Data entry form mirroring the widely used Report and Requisition (R&R) paper form. Data entry by facilities is possible through the desktop browser or a mobile app, including in offline mode. In online mode the form can calculate requisition proposals, offering the facility manager to modify the request and comment on it. These electronic forms can be filled by staff based on the paper stock cards, that are normally placed next to the commodity in the store room.
- DHIS2 can then produce reports for central decision making, giving commodity and program managers the possibility to accept or modify delivery suggestions.
- Stock data can be transformed into logistics indicators, that can be put into context with other program indicators, for example cross-referencing number of patients treated with a specific pathology and corresponding drug consumption.

c) Interoperability Options

LMIS is an area where a multitude of parallel, overlapping or competing software solutions can be found in a single country. As identified in a JSI study in 2012(Ghana Ministry of Health, July 2013: Landscape Analysis of Supply Chain Management Tools in Use), eighteen (18!) different software tools were documented as being in use within the public health supply chain in Ghana alone.

Although a basic LMIS configuration based on aggregate data can take you very far, in some cases a transactional LMIS is necessary if you need to track such detailed operations as returns, transfer between facilities, barcode reading, batch and expiry management. Also some

specialized HQ functions such as creating forecasting, replenishment and elaborate control reports are often done in specialized tools.

DHIS2 has integrated aggregate data from external systems such as openLMIS and CommCare through automated data interfaces. As a result, stock data is available in shared dashboards, displaying health service and stock data next to each other.

9 Installation

The installation chapter provides information on how to install DHIS2 in various contexts, including online central server, offline local network, standalone application and self-contained package called DHIS2 Live.

9.1 Introduction

DHIS2 runs on all platforms for which there exists a Java Runtime Environment version 8 or higher, which includes most popular operating systems such as Windows, Linux and Mac. DHIS2 runs on the PostgreSQL database system. DHIS2 is packaged as a standard Java Web Archive (WAR-file) and thus runs on any Servlet containers such as Tomcat and Jetty.

The DHIS2 team recommends Ubuntu 16.04 LTS operating system, PostgreSQL database system and Tomcat Servlet container as the preferred environment for server installations.

This chapter provides a guide for setting up the above technology stack. It should however be read as a guide for getting up and running and not as an exhaustive documentation for the mentioned environment. We refer to the official Ubuntu, PostgreSQL and Tomcat documentation for in-depth reading.

The dhis2-tools Ubuntu package automates many of the tasks described in the guide below and is recommended for most users, especially those who are not familiar with the command line or administration of servers. It is described in detail in a separate chapter in this guide.

9.2 Server specifications

DHIS2 is a database intensive application and requires that your server has an appropriate amount of RAM, number of CPU cores and a fast disk. These recommendations should be considered as rules-of-thumb and not exact measures. DHIS2 scales linearly on the amount of RAM and number of CPU cores so the more you can afford, the better the application will perform.

- RAM: At least 1 GB memory per 1 million captured data records per month or per 1000 concurrent users. At least 4 GB for a small instance, 12 GB for a medium instance.
- CPU cores: 4 CPU cores for a small instance, 8 CPU cores for a medium or large instance.
- Disk: Ideally use an SSD. Otherwise use a 7200 rpm disk. Minimum read speed is 150 Mb/s, 200 Mb/s is good, 350 Mb/s or better is ideal. In terms of disk space, at least 60 GB is reccomended, but will depend entirely on the amount of data which is contained in the data value tables. Analytics tables require a significant amount of disk space. Plan ahead and ensure that your server can be upgraded with more disk space as it becomes needed.

9.3 Software requirements

Later DHIS2 versions require the following software versions to operate.

- Java JRE version 8 or later.
- Any operating system for which a Java JRE version 8 exists.
- PostgreSQL database version 9.6 or later.
- PostGIS database extension version 2.2 or later.
- Tomcat servlet container version 8 or later, or other Servlet API 3.1 compliant servlet containers like Jetty 9.

9.4 Server setup

This section describes how to set up a server instance of DHIS2 on Ubuntu 16.04 64 bit with PostgreSQL as database system and Tomcat as Servlet container. This guide is not meant to be a step-by-step guide per se, but rather to serve as a reference to how DHIS2 can be deployed on a server. There are many possible deployment strategies, which will differ depending on the operating system and database you are using, and other factors. The term *invoke* refers to executing a given command in a terminal.

For a national server the recommended configuration is a quad-core 2 Ghz processor or higher and 12 Gb RAM or higher. Note that a 64 bit operating system is required for utilizing more than 4 Gb of RAM.

For this guide we assume that 8 Gb RAM is allocated for PostgreSQL and 8 GB RAM is allocated for Tomcat/JVM, and that a 64-bit operating system is used. *If you are running a different configuration please adjust the suggested values accordingly!* We recommend that the available memory is split roughly equally between the database and the JVM. Remember to leave some of the physical memory to the operating system for it to perform its tasks, for instance around 2 GB. The steps marked as *optional*, like the step for performance tuning, can be done at a later stage.

9.4.1 Creating a user to run DHIS2

You should create a dedicated user for running DHIS2. **Under no circumstances, should you run the DHIS2 server as a privileged user such as root.** Create a new user called dhis by invoking:

```
sudo useradd -d /home/dhis -m dhis -s /bin/false
```

Then to set the password for your account invoke:

```
sudo passwd dhis
```

Make sure you set a strong password with at least 15 random characters.

9.4.2 Creating the configuration directory

Start by creating a suitable directory for the DHIS2 configuration files. This directory will also be used for apps, files and log files. An example directory could be:

```
mkdir /home/dhis/config
chown dhis:dhis /home/dhis/config
```

DHIS2 will look for an environment variable called *DHIS2_HOME* to locate the DHIS2 configuration directory. This directory will be referred to as *DHIS2_HOME* in this installation guide. We will define the environment variable in a later step in the installation process.

9.4.3 Setting server time zone and locale

It may be necessary to reconfigure the time zone of the server to match the time zone of the location which the DHIS2 server will be covering. If you are using a virtual private server, the default time zone may not correspond to the time zone of your DHIS2 location. You can easily reconfigure the time zone by invoking the below and following the instructions.

```
sudo dpkg-reconfigure tzdata
```

PostgreSQL is sensitive to locales so you might have to install your locale first. To check existing locales and install new ones (e.g. Norwegian):

```
locale -a
sudo locale-gen nb_NO.UTF-8
```

9.4.4 PostgreSQL installation

Install PostgreSQL by invoking:

```
sudo apt-get install postgresql-10 postgresql-contrib-10 postgresql-10-postgis-2.4
```

Create a non-privileged user called *dhis* by invoking:

```
sudo -u postgres createuser -SDRP dhis
```

Enter a secure password at the prompt. Create a database by invoking:

```
sudo -u postgres createdb -O dhis dhis2
```

Return to your session by invoking `exit` You now have a PostgreSQL user called *dhis* and a database called *dhis2*.

The *PostGIS* extension is needed for several GIS/mapping features to work. DHIS 2 will attempt to install the PostGIS extension during startup. If the DHIS 2 database user does not have permission to create extensions you can create it from the console using the *postgres* user with the following commands:

```
sudo -u postgres psql -c "create extension postgis;" dhis
```

Exit the console and return to your previous user with *\q* followed by *exit*.

9.4.5 PostgreSQL performance tuning

Tuning PostgreSQL is necessary to achieve a high-performing system but is optional in terms of getting DHIS2 to run. PostgreSQL is configured and tuned through the *postgresql.conf* file which can be edited like this:

```
sudo nano /etc/postgresql/9.5/main/postgresql.conf
```

and set the following properties:

```
max_connections = 200
```

Determines maximum number of connections which PostgreSQL will allow.

```
shared_buffers = 3200MB
```

Determines how much memory should be allocated exclusively for PostgreSQL caching. This setting controls the size of the kernel shared memory which should be reserved for PostgreSQL. Should be set to around 40% of total memory dedicated for PostgreSQL.

```
work_mem = 20MB
```

Determines the amount of memory used for internal sort and hash operations. This setting is per connection, per query so a lot of memory may be consumed if raising this too high. Setting this value correctly is essential for DHIS2 aggregation performance.

```
maintenance_work_mem = 512MB
```

Determines the amount of memory PostgreSQL can use for maintenance operations such as creating indexes, running vacuum, adding foreign keys. Incresing this value might improve performance of index creation during the analytics generation processes.

```
effective_cache_size = 8000MB
```

An estimate of how much memory is available for disk caching by the operating system (not an allocation) and isdb.no used by PostgreSQL to determine whether a query plan will fit into memory or not. Setting it to a higher value than what is really available will result in poor performance. This value should be inclusive of the shared_buffers setting. PostgreSQL has two layers of caching: The first layer uses the kernel shared memory and is controlled by the shared_buffers setting. PostgreSQL delegates the second layer to the operating system disk cache and the size of available memory can be given with the effective_cache_size setting.

```
checkpoint_completion_target = 0.8
```

Sets the memory used for buffering during the WAL write process. Increasing this value might improve throughput in write-heavy systems.

```
synchronous_commit = off
```

Specifies whether transaction commits will wait for WAL records to be written to the disk before returning to the client or not. Setting this to off will improve performance considerably. It also implies that there is a slight delay between the transaction is reported successful to the client and it actually being safe, but the database state cannot be corrupted and this is a good alternative for performance-intensive and write-heavy systems like DHIS2.

```
wal_writer_delay = 10000ms
```

Specifies the delay between WAL write operations. Setting this to a high value will improve performance on write-heavy systems since potentially many write operations can be executed within a single flush to disk.

```
random_page_cost = 1.1
```

SSD only. Sets the query planner's estimate of the cost of a non-sequentially-fetched disk page. A low value will cause the system to prefer index scans over sequential scans. A low value makes sense for databases running on SSDs or being heavily cached in memory. The default value is 4.0 which is reasonable for traditional disks.

Restart PostgreSQL by invoking `sudo /etc/init.d/postgresql restart`

9.4.6 Database configuration

The database connection information is provided to DHIS2 through a configuration file called *dhis.conf*. Create this file and save it in the *DHIS2_HOME* directory. As an example this location could be:

```
sudo -u dhis nano /home/dhis/config/dhis.conf
```

A configuration file for PostgreSQL corresponding to the above setup has these properties:

```
# Hibernate SQL dialect
connection.dialect = org.hibernate.dialect.PostgreSQLDialect

# JDBC driver class
connection.driver_class = org.postgresql.Driver

# Database connection URL
connection.url = jdbc:postgresql:dhis2

# Database username
connection.username = dhis

# Database password
connection.password = xxxx

# Database schema behavior, can be validate, update, create, create-drop
connection.schema = update

# Encryption password (sensitive)
encryption.password = xxxx
```

The *encryption.password* property is the password used when encrypting and decrypting data in the database. Note that the password must not be changed once it has been set and data has been encrypted as the data can then no longer be decrypted. Remember to set a strong password of at least **24 characters**.

Note that the configuration file supports environment variables. This means that you can set certain properties as environment variables and have them resolved by DHIS 2, e.g. like this where *DB_PASSWD* is the name of the environment variable:

```
connection.password = ${DB_PASSWD}
```

A common mistake is to have a white-space after the last property value so make sure there is no white-space at the end of any line. Also remember that this file contains the clear text password for your DHIS2 database so it needs to be protected from unauthorized access. To do this invoke the following command which ensures that only the dhis user which owns the file is allowed to read it:

```
chmod 0600 dhis.conf
```

9.4.7 Java installation

Oracle Java 8 JDK is the recommended Java option as it provides the greatest operating system support including Ubuntu LTS 14.04. The *webupd8team Java PPA* provides the necessary packages.

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer
```

Check that your installation is okay by invoking:

```
java -version
```

You can also ensure that the appropriate environment variables are set by installing this package:

```
sudo apt-get install oracle-java8-set-default
```

9.4.8 Tomcat and DHIS2 installation

To install the Tomcat servlet container we will utilize the Tomcat user package by invoking:

```
sudo apt-get install tomcat7-user
```

This package lets us easily create a new Tomcat instance. The instance will be created in the current directory. An appropriate location is the home directory of the dhis user:

```
cd /home/dhis/
sudo tomcat7-instance-create tomcat-dhis
sudo chown -R test:test test-dhis/
```

This will create an instance in a directory called *tomcat-dhis*. Note that the tomcat7-user package allows for creating any number of dhis instances if that is desired.

Next edit the file *tomcat-dhis/bin/setenv.sh* and add the lines below. The first line will set the location of your Java Runtime Environment, the second will dedicate memory to Tomcat and the third will set the location for where DHIS2 will search for the *dhis.conf* configuration file. Please check that the path the Java binaries are correct as they might vary from system to system, e.g. on AMD systems you might see */java-7-openjdk-amd64* Note that you should adjust this to your environment:

```
export JAVA_HOME='/usr/lib/jvm/java-8-oracle/'
export JAVA_OPTS='-Xmx7500m -Xms4000m'
export DHIS2_HOME='/home/dhis/config'
```

The Tomcat configiration file is located in *tomcat-dhis/conf/server.xml*. The element which defines the connection to DHIS is the *Connector* element with port 8080. You can change the port number in the Connector element to a desired port if necessary. If UTF-8 encoding of request data is needed, make sure that the *URIEncoding* attribute is set to *UTF-8*.

```
<Connector port="8080" protocol="HTTP/1.1"
  connectionTimeout="20000"
  redirectPort="8443"
  URIEncoding="UTF-8" />
```

The next step is to download the DHIS2 WAR file and place it into the webapps directory of Tomcat. You can download the DHIS2 version 2.23 WAR release like this (replace 2.23 with your preferred version if necessary):

```
wget https://www.dhis2.org/download/releases/2.26/dhis.war
```

Move the WAR file into the Tomcat webapps directory. We want to call the WAR file ROOT.war in order to make it available at localhost directly without a context path:

```
mv dhis.war tomcat-dhis/webapps/ROOT.war
```

DHIS2 should never be run as a privileged user. After you have modified the setenv.sh file, modify the startup script to check and see if the script has been invoked as root.

```
#!/bin/sh
set -e

if [ "$(id -u)" -eq "0" ]; then
  echo "This script must NOT be run as root" 1>&2
  exit 1
fi

export CATALINA_BASE="/home/dhis/tomcat-dhis"
/usr/share/tomcat7/bin/startup.sh
echo "Tomcat started"
```

9.4.9 Running DHIS2

DHIS2 can now be started by invoking:

```
sudo -u dhis tomcat-dhis/bin/startup.sh
```

> **Warning**
>
> The DHIS2 server should never be run as root or other privileged user.

DHIS2 can be stopped by invoking:

```
sudo -u dhis tomcat-dhis/bin/shutdown.sh
```

To monitor the behavior of Tomcat the log is the primary source of information. The log can be viewed with the following command:

```
tail -f tomcat-dhis/logs/catalina.out
```

Assuming that the WAR file is called ROOT.war, you can now access your DHIS2 instance at the following URL:

```
http://localhost:8080
```

9.5 File store configuration

DHIS2 is capable of capturing and storing files. By default files will be stored on the file system of the server which runs DHIS2 in a *files* directory under the *DHIS2_HOME* external directory location.

You can also configure DHIS2 to store files on cloud-based storage providers. Currently, AWS S3 is the only supported provider. To enable cloud-based storage you must define the following addtional properties in your *dhis.conf* file:

```
# File store provider. Currently 'filesystem' and 'aws-s3' are supported.
filestore.provider = filesystem

# Directory / bucket name. Refers to subdirectory in external directory on file system and
bucket on AWS S3.
filestore.container = files

# The following configuration is applicable only on non-filesystem providers (AWS S3)

# Datacenter location. Not required but recommended for performance reasons.
filestore.location = eu-west-1

# Public identity / username
filestore.identity = xxxx

# Secret password (sensitive)
filestore.secret = xxxx
```

This configuration is an example reflecting the defaults and should be changed to fit your needs. In other words, you can omit it entirely if you plan to use the default values. If you want to use an external provider the last block of properties need to be defined, as well as the *provider* property being set to a supported provider (currently only AWS S3).

> **Note**
>
> If you've configured cloud storage in dhis.conf, all files you upload or the files the system generates will use cloud storage.

For a production system the initial setup of the file store should be carefully considered as moving files across storage providers while keeping the integrity of the database references could be complex. Keep in mind that the contents of the file store might contain both sensitive and integral information and protecting access to the folder as well as making sure a backup plan is in place is recommended on a production implementation.

> **Note**
>
> AWS S3 is the only supported provider (starting from version 2.27) but more providers are likely to be added, such as Google Cloud Store and Rackspace Cloud Files. Let the developers know if you have inquiries about adding support for more providers.

9.6 Google service account configuration

DHIS2 can connect to various Google service APIs. For instance, the DHIS2 GIS component can utilize the Google Earth Engine API to load map layers. In order to provide API access tokens you must set up a Google service account and create a private key:

- Create a Google service account. Please consult the Google identify platform documentation.
- Visit the Google cloud console and go to API Manager > Credentials > Create credentials > Service account key. Select your service account and JSON as key type and click Create.
- Rename the JSON key to *dhis-google-auth.json*.

After downloading the key file, put the *dhis-google-auth.json* file in the DHIS2_HOME directory (the same location as the *dhis.conf* file). As an example this location could be:

```
/home/dhis/config/dhis-google-auth.json
```

9.7 LDAP configuration

DHIS2 is capable of using an LDAP server for authentication of users. For LDAP authentication it is required to have a matching user in the DHIS2 database per LDAP entry. The DHIS2 user will be used to represent authorities / user roles.

To set up LDAP authentication you need to configure the LDAP server URL, a manager user and an LDAP search base and search filter. This configuration should be done in the main DHIS 2 configuration file (dhis.conf). LDAP users, or entries, are identified by distinguished names (DN from now on). An example configuration looks like this:

```
# LDAP server URL
ldap.url = ldaps://domain.org:636

# LDAP manager entry distinguished name
ldap.manager.dn = cn=johndoe,dc=domain,dc=org

# LDAP manager entry password
ldap.manager.password = xxxx

# LDAP base search
ldap.search.base = dc=domain,dc=org
```

```
# LDAP search filter
ldap.search.filter = (cn={0})
```

The LDAP configuration properties are explained below:

- *ldap.url:* The URL of the LDAP server for which to authenticate against. Using SSL/ encryption is strongly recommended in order to make authentication secure. As example URL is *ldaps://domain.org:636*, where ldaps refers to the protocol, *domain.org* refers to the domain name or IP address, and *636* refers to the port (636 is default for LDAPS).
- *ldap.manager.dn:* An LDAP manager user is required for binding to the LDAP server for the user authentication process. This property refers to the DN of that entry. I.e. this is not the user which will be authenticated when logging into DHIS2, rather the user which binds to the LDAP server in order to do the authentication.
- *ldap.manager.password:* The password for the LDAP manager user.
- *ldap.search.base:* The search base, or the distinguished name of the search base object, which defines the location in the directory from which the LDAP search begins.
- *ldap.search.filter:* The filter for matching DNs of entries in the LDAP directory. The {0} variable will be substituted by the DHIS2 username, or alternatively, the LDAP identifier defined for the user with the supplied username.

DHIS2 will use the supplied username / password and try to authenticate against an LDAP server entry, then look up user roles / authorities from a corresponding DHIS2 user. This implies that a user must have a matching entry in the LDAP directory as well as a DHIS2 user in order to log in.

During authentication, DHIS2 will try to bind to the LDAP server using the configured LDAP server URL and the manager DN and password. Once the binding is done, it will search for an entry in the directory using the configured LDAP search base and search filter.

The {0} variable in the configured filter will be substituted beforing applying the filter. By default, it will be substituted by the supplied username. You can also set a custom LDAP identifier on the relevant DHIS2 user account. This can be done throught the DHIS2 user module user interface in the add or edit screen by setting the “LDAP identifier” property. When set, the LDAP identifier will be subsituted for the {0} variable in the filter. This feature is useful when the LDAP common name is not suitable or cannot for some reason be used as a DHIS2 username.

9.8 Encryption configuration

DHIS2 allows for encryption of data. This however requires some extra setup.

9.8.1 Java Cryptography Extension

DHIS2 uses an encryption algorithm classified as strong and therefore requires the *Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files* to be installed. These files can be installed through these steps:

1. Download the JCE Unlimited Strength Jurisdiction Policy Files for your java version of Java from the Oracle Web site. Scroll down to the “Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files” section. It is important that the version of the files match the version of Java on your server.

   http://www.oracle.com/technetwork/java/javase/downloads/index.html

2. Extract the downloaded ZIP archive. It contains two JAR files: *local_policy.jar* and *US_export_policy.jar*.

3. Locate the JDK directory of your Java installation. From there, navigate into the *jre/security* directory. On Ubuntu it is often found at */usr/lib/jvm/java-8-oracle/jre/lib/security*.

4. (Optional) Back up your existing *local_policy.jar* and *US_export_policy.jar* in case you want to revert to them later.

5. Copy the *local_policy.jar* and *US_export_policy.jar* files into the security folder. You should now have the following files which completes the installation. Remember to restart your servlet container for it to take effect.

```
/usr/lib/jvm/java-8-oracle/jre/lib/security/local_policy.jar
/usr/lib/jvm/java-8-oracle/jre/lib/security/US_export_policy.jar
```

9.8.2 Password configuration

To provide security to the encryption alogorithm you will have to set a password in the *dhis.conf* configuration file through the *encryption.password* property:

```
encryption.password = xxxx
```

The password must be at least **24 characters long** and it is recommended to use a mix of numbers and lower- and uppercase letters. The encryption password must be kept secret.

9.8.3 Considerations for encryption

A word of caution: It is not possible to recover encrypted data if the encryption password is lost or changed. Conversely, the encryption provides no security if the password is compromised. Hence, great consideration should be given to storing the password in a safe place.

9.9 Read replica database configuration

DHIS 2 allows for utilizing read only replicas of the master database (the main DHIS 2 database). The purpose of read replicas is to enhance the performance of database read queries and scale out the capacity beyond the constraints of a single database. Read-heavy operations such as analytics and event queries will benefit from this.

The configuration requires that you have created one or more replicated instances of the master DHIS 2 database. PostgreSQL achieves this through a concept referred to as *streaming replication*. Configuring read replicas for PostgreSQL is not covered in this guide.

Read replicas can be defined in the *dhis.conf* configuration file. You can specify up to 5 read replicas per DHIS 2 instance. Each read replica is denoted with a number between 1 and 5. The JDBC connection URL must be defined per replica. The username and password can be specified; if not, the username and password for the master database will be used instead.

The configuration for read replicas in *dhis.conf* looks like the below. Each replica is specified with the configuration key *readN* prefix, where N refers to the replica number.

```
# Read replica 1 configuration

# Database connection URL, username and password
read1.connection.url = jdbc:postgresql://127.0.0.11/dbread1
read1.connection.username = dhis
read1.connection.password = xxxx
```

```
# Read replica 2 configuration

# Database connection URL, username and password
read2.connection.url = jdbc:postgresql://127.0.0.12/dbread2
read2.connection.username = dhis
read2.connection.password = xxxx

# Read replica 3 configuration

# Database connection URL, fallback to master for username and password
read3.connection.url = jdbc:postgresql://127.0.0.13/dbread3
```

Note that you must restart your servlet container for the changes to take effect. DHIS 2 will automatically distribute the load across the read replicas. The ordering of replicas has no significance.

9.10 Web server cluster configuration

This section describes how to set up the DHIS 2 application to run in a cluster.

9.10.1 Clustering overview

Clustering is a common technique for improving system scalability and availability. Clustering refers to setting up multiple web servers such as Tomcat instances and have them serve a single application. Clustering allows for *scaling out* an application in the sense that new servers can be added to improve performance. It also allows for *high availability* as the system can tolerate instances going down without making the system inaccessible to users.

When setting up multiple Tomcat instances there is a need for making the instances aware of each other. This awareness will enable DHIS 2 to keep the local data (Hibernate) caches in sync and in a consistent state. When an update is done on one instance, the caches on the other instances must be notified so that they can be invalidated and avoid becoming stale.

There are two aspects to configure in *dhis.conf* in order to run DHIS 2 in a cluster.

- Each instance must specify the other DHIS 2 application members of the cluster.
- An instance of the Redis data store must be installed and connection information must be configured for each DHIS 2 application instance.

9.10.2 Cluster instance configuration

A DHIS 2 cluster setup is based on manual configuration of each instance. For each DHIS 2 instance one must specify the public *hostname* as well as the hostnames of the other instances participating in the cluster.

The hostname of the server is specified using the *cluster.hostname* configuration property. Additional servers which participate in the cluster are specified using the *cluster.members* configuration property. The property expects a list of comma separated values where each value is of the format *host:port*.

The hostname must be visible to the participating servers on the network for the clustering to work. You might have to allow incoming and outgoing connections on the configured port numbers in the firewall.

The port number of the server is specified using the*cluster.cache.port* configuration property. The remote object port used for registry receive calls is specified using *cluster.cache.remote.object.port*. Specifying the port numbers is typically only useful when you have multiple cluster instances on the same server / virtual machine or if you need to explicitly

specify the ports to be used so as to have them configured in firewall. When running cluster instances on separate servers / virtual machines it is often appropriate to use the default port number and omit the ports configuration properties. If omitted, 4001 will be assigned as the listener port and a random free port will be assigned as the remote object port.

An example setup for a cluster of two web servers is described below. For *server A* available at hostname *193.157.199.131* the following can be specified in *dhis.conf*:

```
# Cluster configuration for server A

# Hostname for this web server
cluster.hostname = 193.157.199.131

# Ports for cache listener, can be omitted
cluster.cache.port = 4001
cluster.cache.remote.object.port = 5001

# List of Host:port participating in the cluster
cluster.members = 193.157.199.132:4001
```

For *server B* available at hostname *193.157.199.132* the following can be specified in *dhis.conf* (notice how port configuration is omitted):

```
# Cluster configuration for server B

# Hostname for this web server
cluster.hostname = 193.157.199.132

# List of servers participating in cluster
cluster.members = 193.157.199.131:4001
```

You must restart each Tomcat instance to make the changes take effect. The two instances have now been made aware of each other and DHIS 2 will ensure that their caches are kept in sync.

9.10.3 Cluster shared data store configuration

In a cluster setup, a *Redis* instance is required and will handle shared user sessions, application cache and cluster node leadership.

For optimum performance, *Redis Keyspace events* for Generic commands and Expired events needs to be enabled in your Redis Server. If you are using any cloud platform managed Redis server (like AWS ElastiCache for Redis or Azure Cache for Redis), you will have to enable keyspace event notifications using the respective cloud interfaces. If you are setting up a standalone Redis server, enabling keyspace event notifications can be done in the *redis.conf* file by adding/ uncommenting the following line

```
notify-keyspace-events Egx
```

DHIS2 will connect to Redis if the *redis.enabled* configuration property in *dhis.conf* is set to *true* along with the following four properties:

1. *redis.host*: Specifies where the redis server is running. Defaults to *localhost*.
2. *redis.port*: Specifies the port in which the redis server is listening. Defaults to *6379*.

3. *redis.password*: Specifies the authentication password.
4. *redis.use.ssl*: Specifies whether the Redis server has SSL enabled. Defaults to *false*.

Whenever Redis is enabled, DHIS2 will automatically assign one of the running instances as the leader of the cluster. The leader instance will be used to execute jobs or scheduled tasks that should be run exclusively by one instance. Optionally, you can configure the *leader.time.to.live.minutes* property in *dhis.conf* to set up how frequently the leader election needs to occur. It also gives an indication of how long it would take for another instance to take over as the leader after the previous leader has shutdown/crashed. The default value is 2 minutes. Note that assigning a leader in the cluster is only done if Redis is enabled. An example snippet of the *dhis.conf* configuration file with Redis enabled and leader election time configured is shown below.

```
# Redis Configuration

# Mandatory if redis has to be enabled.
redis.enabled = true

redis.host = 193.158.100.111

redis.port = 6379

redis.password = yourpassword

# Optional, defaults to false.
redis.use.ssl = false

# Optional, defaults to 2 minutes.
leader.time.to.live.minutes=4
```

9.10.4 Load balancing

With a cluster of Tomcat instances set up, a common approach for routing incoming web requests to the backend instances participating in the cluster is using a *load balancer*. A load balancer will make sure that load is distributed evenly across the cluster instances. It will also detect whether an instance becomes unavailable, and if so, stop routine requests to that instance and instead use other available instances.

Load balancing can be achieved in multiple ways. A simple approach is using *nginx*, in which case you will define an *upstream* element which enumerates the location of the backend instances and later use that element in the *proxy* location block.

```
http {

  # Upstream element with sticky sessions

  upstream dhis_cluster {
    ip_hash;
    server 193.157.199.131:8080;
    server 193.157.199.132:8080;
  }

  # Proxy pass to backend servers in cluster

  server {
```

```
    listen 80;

    location / {
      proxy_pass   http://dhis_cluster/;
    }
  }
}
```

DHIS 2 keeps server-side state for user sessions to a limited degree. Using “sticky sessions” is a simple approach to avoid replicating the server session state by routing requests from the same client to the same server. The *ip_hash* directive in the upstream element ensures this.

Note that several instructions have been omitted for brevity in the above example. Consult the reverse proxy section for a detailed configuration guide.

9.11 Starting Tomcat at boot time

In certain situations a server might reboot unexpectedly. It is hence preferable to have Tomcat start automatically when the server starts. To achieve that the first step is to create init scripts. Create a new file called `tomcat` and paste the below content into it (adjust the HOME variable to your environment):

```
#!/bin/sh
#Tomcat init script

HOME=/home/dhis/tomcat/bin

case $1 in
start)
        sh ${HOME}/startup.sh
        ;;
stop)
        sh ${HOME}/shutdown.sh
        ;;
restart)
        sh ${HOME}/shutdown.sh
        sleep 5
        sh ${HOME}/startup.sh
        ;;
esac
exit 0
```

Move the script to the init script directory and make them executable by invoking:

```
sudo mv tomcat /etc/init.d
sudo chmod +x /etc/init.d/tomcat
```

Next make sure the tomcat init script will be invoked during system startup and shutdown:

```
sudo /usr/sbin/update-rc.d -f tomcat defaults 81
```

Tomcat will now be started at system startup and stopped at system shutdown. If you later need to revert this you can replace `defaults` with `remove` and invoke the above commands again.

9.12 Reverse proxy configuration

A reverse proxy is a proxy server that acts on behalf of a server. Using a reverse proxy in combination with a servlet container is optional but has many advantages:

- Requests can be mapped and passed on to multiple servlet containers
  - this improves flexibility and makes it easier to run multiple instances of DHIS2 on the same server. It also makes it possible to change the internal server setup without affecting clients.
- The DHIS2 application can be run as a non-root user on a port different than 80 which reduces the consequences of session hijacking.
- The reverse proxy can act as a single SSL server and be configured to inspect requests for malicious content, log requests and responses and provide non-sensitive error messages which will improve security.

9.12.1 Basic nginx setup

We recommend using nginx as reverse proxy due to its low memory footprint and ease of use. To install invoke the following:

```
sudo apt-get install nginx
```

nginx can now be started, reloaded and stopped with the following commands:

```
sudo /etc/init.d/nginx start
sudo /etc/init.d/nginx reload
sudo /etc/init.d/nginx stop
```

Now that we have installed nginx we will now continue to configure regular proxying of requests to our Tomcat instance, which we assume runs at *http://localhost:8080*. To configure nginx you can open the configuration file by invoking:

```
sudo nano /etc/nginx/nginx.conf
```

nginx configuration is built around a hierarchy of blocks representing http, server and location, where each block inherit settings from parent blocks. The following snippet will configure nginx to proxy pass (redirect) requests from port 80 (which is the port nginx will listen on by default) to our Tomcat instance. Include the following configuration in nginx.conf:

```
http {
  gzip on; # Enables compression, incl Web API content-types
  gzip_types
    "application/json;charset=utf-8" application/json
    "application/javascript;charset=utf-8" application/javascript text/javascript
    "application/xml;charset=utf-8" application/xml text/xml
    "text/css;charset=utf-8" text/css
    "text/plain;charset=utf-8" text/plain;

  server {
    listen               80;
    root  /home/dhis/tomcat/webapps/ROOT; # Update path!
```

```
    client_max_body_size 10M;

    # Serve static files

    location ~ (\.js|\.css|\.gif|\.woff|\.ttf|\.eot|\.ico|(/dhis-web-commons/|/images/|/
icons/).*\.png)$ {
      add_header  Cache-Control public;
      expires     14d;
    }

    # Proxy pass to servlet container

    location / {
      proxy_pass                http://localhost:8080/;
      proxy_redirect            off;
      proxy_set_header          Host               $host;
      proxy_set_header          X-Real-IP          $remote_addr;
      proxy_set_header          X-Forwarded-For    $proxy_add_x_forwarded_for;
      proxy_set_header          X-Forwarded-Proto  http;
      proxy_buffer_size         128k;
      proxy_buffers             8 128k;
      proxy_busy_buffers_size   256k;
    }
  }
}
```

You can now access your DHIS2 instance at *http://localhost*. Since the reverse proxy has been set up we can improve security by making Tomcat only listen for local connections. In */conf/server.xml* you can add an *address* attribute with the value *localhost* to the Connector element for HTTP 1.1 like this:

```
<Connector address="localhost" protocol="HTTP/1.1" ... >
```

9.12.2 Enabling SSL on nginx

In order to improve security it is recommended to configure the server running DHIS2 to communicate with clients over an encrypted connection and to identify itself to clients using a trusted certificate. This can be achieved through SSL which is an cryptographic communication protocol running on top of TCP/IP. First, install the required *openssl* library:

```
sudo apt-get install openssl
```

To configure nginx to use SSL you will need a proper SSL certificate from an SSL provider. The cost of a certificate varies a lot depending on encryption strength. An affordable certificate from Rapid SSL Online should serve most purposes. To generate the CSR (certificate signing request) you can invoke the command below. When you are prompted for the *Common Name*, enter the fully qualified domain name for the site you are securing.

```
openssl req -new -newkey rsa:2048 -nodes -keyout server.key -out server.csr
```

When you have received your certificate files (.pem or .crt) you will need to place it together with the generated server.key file in a location which is reachable by nginx. A good location for this can be the same directory as where your nginx.conf file is located.

Below is an nginx server block where the certificate files are named server.crt and server.key. Since SSL connections usually occur on port 443 (HTTPS) we pass requests on that port (443) on to the DHIS2 instance running on *http://localhost:8080* The first server block will rewrite all requests connecting to port 80 and force the use of HTTPS/SSL. This is also necessary because DHIS2 is using a lot of redirects internally which must be passed on to use HTTPS. Remember to replace *<server-ip>* with the IP of your server. These blocks should replace the one from the previous section.

```
http {
  gzip on; # Enables compression, incl Web API content-types
  gzip_types
    "application/json;charset=utf-8" application/json
    "application/javascript;charset=utf-8" application/javascript text/javascript
    "application/xml;charset=utf-8" application/xml text/xml
    "text/css;charset=utf-8" text/css
    "text/plain;charset=utf-8" text/plain;

  # HTTP server - rewrite to force use of SSL

  server {
    listen     80;
    rewrite    ^ https://<server-url>$request_uri? permanent;
  }

  # HTTPS server

  server {
    listen               443 ssl;
    root  /home/dhis/tomcat/webapps/ROOT; # Update path!
    client_max_body_size 10M;

    ssl                  on;
    ssl_certificate      server.crt;
    ssl_certificate_key  server.key;

    ssl_session_cache    shared:SSL:20m;
    ssl_session_timeout  10m;

    ssl_protocols              TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers                RC4:HIGH:!aNULL:!MD5;
    ssl_prefer_server_ciphers  on;

    # Serve static files

    location ~ (\.js|\.css|\.gif|\.woff|\.ttf|\.eot|\.ico|(/dhis-web-commons/|/images/|/
icons/).*\.png)$ {
      add_header  Cache-Control public;
      expires     14d;
    }

    # Proxy pass to servlet container

    location / {
      proxy_pass               http://localhost:8080/;
      proxy_redirect           off;
      proxy_set_header         Host               $host;
      proxy_set_header         X-Real-IP          $remote_addr;
      proxy_set_header         X-Forwarded-For    $proxy_add_x_forwarded_for;
      proxy_set_header         X-Forwarded-Proto  https;
      proxy_buffer_size        128k;
```

```
      proxy_buffers             8 128k;
      proxy_busy_buffers_size   256k;
    }
  }
}
```

Note the last “https” header value which is required to inform the servlet container that the request is coming over HTTPS. In order for tomcat to properly produce Location URLs using https you also need to add two other parameters to the Connector in tomcat’s server.xml file:

```
<Connector scheme="https" proxyPort="443" ... >
```

9.12.3 Enabling caching and SSL on nginx

Requests for reports, charts, maps and other analysis-related resources will often take some time to respond and might utilize a lot of server resources. In order to improve response times, reduce the load on the server and hide potential server downtime we can introduce a cache proxy in our server setup. The cached content will be stored in directory /var/cache/nginx, and up to 250 MB of storage will be allocated. Nginx will create this directory automatically.

```
http {
  # ...
  root              /home/dhis/tomcat/webapps/ROOT; # Update path!
  proxy_cache_path  /var/cache/nginx  levels=1:2  keys_zone=dhis:250m  inactive=1d;

  gzip on; # Enables compression, incl Web API content-types
  gzip_types
    "application/json;charset=utf-8" application/json
    "application/javascript;charset=utf-8" application/javascript text/javascript
    "application/xml;charset=utf-8" application/xml text/xml
    "text/css;charset=utf-8" text/css
    "text/plain;charset=utf-8" text/plain;

  # HTTP server - rewrite to force use of HTTPS

  server {
    listen     80;
    rewrite    ^ https://www.domain.com/$request_uri? permanent;
  }

  # HTTPS server

  server {
    listen               443 ssl;
    client_max_body_size 10M;

    ssl                  on;
    ssl_certificate      server.crt;
    ssl_certificate_key  server.key;

    ssl_session_timeout  30m;

    ssl_protocols              SSLv2 SSLv3 TLSv1;
    ssl_ciphers                HIGH:!aNULL:!MD5;
    ssl_prefer_server_ciphers  on;

    # Serve static files
```

```
    location ~ (\.js|\.css|\.gif|\.woff|\.ttf|\.eot|\.ico|(/dhis-web-commons/|/images/|/
icons/).*\.png)$ {
      add_header  Cache-Control public;
      expires     14d;
    }

    # Proxy pass to servlet container and potentially cache response

    location / {
      proxy_pass                http://localhost:8080/;
      proxy_redirect            off;
      proxy_set_header          Host               $host;
      proxy_set_header          X-Real-IP          $remote_addr;
      proxy_set_header          X-Forwarded-For    $proxy_add_x_forwarded_for;
      proxy_set_header          X-Forwarded-Proto  https;
      proxy_buffer_size         128k;
      proxy_buffers             8 128k;
      proxy_busy_buffers_size   256k;
      proxy_cache               dhis;
    }
  }
}
```

> **Important**
>
> Be aware that a server side cache shortcuts the DHIS2 security features in the sense that requests which hit the server side cache will be served directly from the cache outside the control of DHIS2 and the servlet container. This implies that request URLs can be guessed and reports retrieved from the cache by unauthorized users. Hence, if you capture sensitive information, setting up a server side cache is not recommended.

9.12.4 Additional resources on SSL

The configuration demonstrated above should be regarded as the absolute minumum in order to establish a secure server. However, encryption methods are constantly being updated, so implementers who are administerting their own server, show ensure that the server is regularly updated with recent security patches (particularly the HTTP server and SSL libraries).

There are numerous additional tutorials and information available on the web, including a helpful step-by-step guide for using the free Lets Encrypt SSL certifcate system . It may also be useful to regularly test your SSL security with this website.

9.12.5 Making resources available with nginx

In some scenarios it is desirable to make certain resources publicly available on the Web without requiring authentication. One example is when you want to make data analysis related resources in the Web API available in a Web portal. The following example will allow access to charts, maps, reports, report table and document resources through basic authentication by injecting an *Authorization* HTTP header into the request. It will remove the Cookie header from the request and the Set-Cookie header from the response in order to avoid changing the currently logged in user. It is recommended to create a user for this purpose given only the minimum authorities required. The Authorization value can be constructed by Base64-encoding the username appended with a colon and the password and prefix it “Basic”, more precisely “Basic base64_encode(username:password)”. It will check the HTTP method used for requests and return *405 Method Not Allowed* if anything but GET is detected.

It can be favorable to set up a separate domain for such public users when using this approach. This is because we don't want to change the credentials for already logged in users when they access the public resources. For instance, when your server is deployed at somedomain.com, you can set a dedicated subdomain at api.somedomain.com, and point URLs from your portal to this subdomain.

```
server {
  listen       80;
  server_name  api.somedomain.com;

  location ~ ^/(api/(charts|chartValues|reports|reportTables|documents|maps|organisationUnits)|
dhis-web-commons/javascripts|images|dhis-web-commons-ajax-json|dhis-web-mapping|dhis-web-
visualizer) {
    if ($request_method != GET) {
      return 405;
    }

    proxy_pass         http://localhost:8080;
    proxy_redirect     off;
    proxy_set_header   Host               $host;
    proxy_set_header   X-Real-IP          $remote_addr;
    proxy_set_header   X-Forwarded-For    $proxy_add_x_forwarded_for;
    proxy_set_header   X-Forwarded-Proto  http;
    proxy_set_header   Authorization      "Basic YWRtaW46ZGlzdHJpY3Q=";
    proxy_set_header   Cookie             "";
    proxy_hide_header  Set-Cookie;
  }
}
```

9.12.6 Basic reverse proxy setup with Apache

The Apache HTTP server is the most common

> **Important**
>
> Using nginx is the preferred option as reverse proxy with DHIS2 and you should not attempt to install both nginx and Apache on the same server. If you have installed nginx please ignore this section.

The Apache HTTP server is the most widely used HTTP server currently. Depdenign on your exact nature of deployment, you may need to use Apache as a reverse proxy for your DHIS2 server. In this section, we will describe how to implement a simple reverse proxy setup with Apache.

First we need to install a few necessary programs modules for Apache and enable the modules.

```
sudo apt-get install apache2 libapache2-mod-proxy-html libapache2-mod-jk
a2enmod proxy proxy_ajp proxy_connect
```

Lets define an AJP connector which Apache HTTP server will use to connect to Tomcat with. The Tomcat `server.xml` file should be located in the /conf/ director of your Tomcat installation. Be sure this line is uncommented.You can set the port to anything you like which is unused.

```
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

Now, we need to make the adjustments to the Apache HTTP server which will answer requests on port 80 and pass them to the Tomcat server through an AJP connector. Edit the file `/etc/apache2/mods-enabled/proxy.conf` so that it looks like the example below. Be sure that the port defined in the configuration file matches the one from Tomcat.

```
<IfModule mod_proxy.c>

ProxyRequests Off
ProxyPass /dhis  ajp://localhost:8009/dhis
ProxyPassReverse /dhis  ajp://localhost:8009/dhis

<Location "/dhis">
  Order allow,deny
  Allow from all
</Location>
</IfModule>
```

You now can restart Tomcat and the Apache HTTPD server and your DHIS2 instance should be available on http://*myserver*/dhis where *myserver* is the hostname of your server.

9.12.7 SSL encryption with Apache

Using Apache and the reverse proxy setup described in the previous section, we can easily implement encrypted transfer of data between clients and the server over HTTPS. This section will describe how to use self-signed certificates, although the same procedure could be used if you have fully-signed certificates as well.

First (as root), generate the necessary private key files and CSR (Certificate Signing Request)

```
mkdir /etc/apache2/ssl
cd /etc/apache2/ssl
openssl genrsa -des3 -out server.key 1024
openssl req -new -key server.key -out server.csr
```

We need to remove the password from the key, otherwise Apache will not be able to use it.

```
cp server.key server.key.org
openssl rsa -in server.key.org -out server.key
```

Next, generate a self-signed certificate which will be valid for one year.

```
openssl x509 -req -days 365 -in server.csr -signkey \ server.key -out server.crt
```

Now, lets configure Apache by enabling the SSL modules and creating a default site.

```
a2enmod ssl
a2ensite default-ssl
```

Now, we need to edit the default-ssl (located at `/etc/apache2/sites-enabled/default-ssl`) file in order to enable the SSL transfer functionality of Apache.

```
<VirtualHost *:443>
        ServerAdmin wemaster@mydomain.org
       SSLEngine On
       SSLCertificateFile /etc/apache2/ssl/server.crt
       SSLCertificateKeyFile /etc/apache2/ssl/server.key
...
```

Be sure that the *:80 section of this file is changed to port *:443, which is the default SSL port. Also, be sure to change the ServerAdmin to the webmaster's email. Lastly, we need to be sure that the hostname is setup properly in /etc/hosts. Just under the "localhost" line, be sure to add the server's IP address and domain name.

```
127.0.0.1 localhost
XXX.XX.XXX.XXX foo.mydomain.org
```

Now, just restart Apache and you should be able to view https://foo.mydomain.org/dhis.

```
/etc/init.d/apache2 restart
```

9.13 DHIS2 configuration reference

The following describes the full set of configuration options for the *dhis.conf* configuration file. The uncommented properties are mandatory. The commented properties are optional. The configuration file should be placed in a directory which is pointed to by a *DHIS2_HOME* environment variable. The comment (#) must be removed for a property value to take effect. You can copy and paste the following content as a viable starting point for your own configuration file.

```
# ------------------------------------------------------------------------
# Database connection for PostgreSQL
# ------------------------------------------------------------------------

# Hibernate SQL dialect
connection.dialect = org.hibernate.dialect.PostgreSQLDialect

# JDBC driver class
connection.driver_class = org.postgresql.Driver

# Database connection URL
connection.url = jdbc:postgresql:dhis2

# Database username
connection.username = dhis

# Database password (sensitive)
connection.password = xxxx

# Database schema behavior, can be 'validate', 'update', 'create', 'create-drop'
connection.schema = update

# Max size of connection pool (default: 40)
# connection.pool.max_size = 40

# ------------------------------------------------------------------------
```

```
# System
# -----------------------------------------------------------------------

# System mode for database read operations only, can be 'off', 'on'
# system.read_only_mode = off

# Session timeout in seconds, default is 3600
# system.session.timeout = 3600

# SQL view protected tables, can be 'on', 'off'
# system.sql_view_table_protection = on

# -----------------------------------------------------------------------
# Encryption
# -----------------------------------------------------------------------

# Encryption password (sensitive)
# encryption.password = xxxx

# -----------------------------------------------------------------------
# File store
# -----------------------------------------------------------------------

# File store provider, currently 'filesystem' and 'aws-s3' are supported
# filestore.provider = filesystem

# Directory / bucket name, refers to folder within DHIS2_HOME on file system, 'bucket' on AWS S3
# filestore.container = files

# Datacenter location (not required)
# filestore.location = eu-west-1

# Public identity / username
# filestore.identity = dhis2-id

# Secret key / password (sensitive)
# filestore.secret = xxxx

# -----------------------------------------------------------------------
# LDAP
# -----------------------------------------------------------------------

# LDAP server URL
# ldap.url = ldaps://300.20.300.20:636

# LDAP manager user distinguished name
# ldap.manager.dn = cn=JohnDoe,ou=Country,ou=Admin,dc=hisp,dc=org

# LDAP manager user password (sensitive)
# ldap.manager.password = xxxx

# LDAP entry distinguished name search base
# ldap.search.base = dc=hisp,dc=org

# LDAP entry distinguished name filter
# ldap.search.filter = (cn={0})

# -----------------------------------------------------------------------
# Node
# -----------------------------------------------------------------------

# Node identifier, optional, useful in clusters
# node.id = 'node-1'
```

```
# ------------------------------------------------------------------------
# System monitoring
# ------------------------------------------------------------------------

# System monitoring URL
# system.monitoring.url =

# System monitoring username
# system.monitoring.username =

# System monitoring password
# system.monitoring.password =
```

9.14 Application logging

This section covers application logging in DHIS 2.

9.14.1 Log files

The DHIS2 application log output is directed to multiple files and locations. First, log output is sent to standard output. The Tomcat servlet container usually outputs standard output to a file under “logs”:

```
<tomcat-dir>/logs/catalina.out
```

Second, log output is written to a “logs” directory under the DHIS2 home directory as defined by the the DHIS2_HOME environment variables. There is a main log file for all output, and separate log files for various background processes. The main file includes the background process logs as well. The log files are capped at 50 Mb and log content is continuously appended.

```
<DHIS2_HOME>/logs/dhis.log
<DHIS2_HOME>/logs/dhis-analytics-table.log
<DHIS2_HOME>/logs/dhis-data-exchange.log
<DHIS2_HOME>/logs/dhis-data-sync.log
```

9.14.2 Log configuration

In order to override the default log configuration you can specify a Java system property with the name *log4j.configuration* and a value pointing to the Log4j configuration file on the classpath. If you want to point to a file on the file system (i.e. outside Tomcat) you can use the *file* prefix e.g. like this:

```
-Dlog4j.configuration=file:/home/dhis/config/log4j.properties
```

Java system properties can be set e.g. through the *JAVA_OPTS* environment variable or in the tomcat startup script.

A second approach to overriding the log configuration is to specify logging properties in the *dhis.conf* configuration file. The supported properties are:

```
# Max size for log files, default is '100MB'
logging.file.max_size = 250MB
```

```
# Max number of rolling log archive files, default is 0
logging.file.max_archives = 2
```

DHIS2 will eventually phase out logging to standard out / catalina.out and as a result it is recommended to rely on the logs under DHIS2_HOME.

9.15 Working with the PostgreSQL database

Common operations when managing a DHIS2 instance are dumping and restoring databases. To make a dump (copy) of your database, assuming the setup from the installation section, you can invoke the following:

```
pg_dump dhis2 -U dhis -f dhis2.sql
```

The first argument (dhis2) refers to the name of the database. The second argument (dhis) refers to the database user. The last argument (dhis2.sql) is the file name of the copy. If you want to compress the file copy immediately you can do:

```
pg_dump dhis2 -U dhis | gzip > dhis2.sql.gz
```

To restore this copy on another system, you first need to create an empty database as described in the installation section. You also need to gunzip the copy if you created a compressed version. You can the invoke:

```
psql -d dhis2 -U dhis -f dhis2.sql
```

9.16 DHIS2 Live setup

The DHIS2 Live package is extremely convenient to install and run. It is intended for demonstrations, for users who want to explore the system and for small, offline installations typically at districts or facilities. It only requires a Java Runtime Environment and runs on all browsers except Internet Explorer 7 and lower.

To install start by downloading DHIS2 Live from *http://dhis2.org* and extract the archive to any location. On Windows click the executable archive. On Linux invoke the startup.sh script. After the startup process is done your default web browser will automtically be pointed to *http://localhost:8082* where the application is accessible. A system tray menu is accessible on most operating systems where you can start and stop the server and start new browser sesssions. Please note that if you have the server running there is no need to start it again, simply open the application from the tray menu.

DHIS2 Live is running on an embedded Jetty servlet container and an embedded H2 database. However it can easily be configured to run on other database systems such as PostgreSQL. Please read the section above about server installations for an explanation of the database configuration. The *dhis.conf* configuration file is located in the *conf* folder. Remember to restart the Live package for your changes to take effect. The server port is 8082 by default. This can be changed by modifying the value in the*jetty.port* configuration file located in the *conf* directory.

10 Support

The DHIS2 community uses a set of collaboration and coordination platforms for information and provision of downloads, documentation, development, source code, functionality specifications, bug tracking. This chapter will describe these in more detail.

10.1 Home page: dhis2.org

The DHIS2 home page is found at *http://dhis2.org*. The *download* page provides downloads for the DHIS2 Live package, WAR files, the mobile client, a Debian package, the source code, sample databases and a tool for editing the application user interface translations. Please note that the current latest release will be maintained until the next is released and both the actual release and the latest build from the release branch are provided. We recommend that you check back regularly on the download page and update your online server with the latest build from the release branch. The build revision can be found under *Help* - *About* inside DHIS2.

The *documentation* page provides installation instructions, user documentation, this implementation guide, presentations, Javadocs, changelog, roadmap and a guide for contributing to the documentation. The user documentation is focused on the practical aspects of using DHIS2, such as how to create data elements and reports. This implementation guide is addressing the more high-level aspects of DHIS2 implementation, database development and maintenance. The change log and roadmap sections provide links to the relevant pages in the Launchpad site described later.

The *functionality* and *features* pages give a brief overview with screenshots of the main functionalities and features of DHIS2. A demo DHIS2 application can be reached from the *demo* top menu link. These pages can be used when a quick introduction to the system must be given to various stakeholders.

The *about* page has information about the license under which DHIS2 is released, how to sign up for the mailing lists, get access to the source code and more.

10.2 Collaboration platform: launchpad.net/dhis2

DHIS2 uses *Launchpad* as the main collaboration platform. The site can be accessed at *http://lanchpad.net/dhis2* Launchpad is used for source code hosting, functionality specifications, bug tracking and notifications. The *Bazaar* version control system is tightly integrated with Launchpad and is required for checking out the source code.

The various source code branches including trunk and release branches can be browsed at *http://code.launchpad.net/dhis2*

If you want to suggest new functionality to be implemented in DHIS2 you can air your views on the developer mailing list and eventually write a specification, which is referred to as *blueprints* in Launchpad. The bueprint will be considered by the core development team and if accepted be assigned a developer, approver and release version. Blueprints can be browsed and added at *http://blueprints.launchpad.net/dhis2*

If you find a bug in DHIS2 you can report it at Launchpad by navigating to *http://bugs.launchpad.net/dhis2* Your bug report will be investigated by the developer team and be given a status. If valid it will also be assigned to a developer and approver and eventually be fixed. Note that bugfixes are incorporated into the trunk and latest release branch - so more testing and feedback to the developer teams leads to higher quality of your software.

10.3 Reporting a problem

If you encounter a problem with the DHIS2 software you can ask for assistance on the developer's mailing list at https://launchpad.net/~dhis2-devs. Before reporting, make sure that you have:

- Cleared the web browser cache (also called history or browsing data) completely (select all options before clearing).
- Cleared the DHIS2 application cache: Go to Data administration -> Cache statistics and click Clear cache.

If you believe you have found a bug you can report it either to the developer mailing lists or to the bug tracker at https://bugs.launchpad.net/dhis2 .

For the developers to be able to help you need to provide as much useful information as possible:

- DHIS2 version: Look in the Help -> About page inside DHIS2 and provide the version and build revision.
- Web browser including version.
- Operating system including version.
- Servlet container / Tomcat log: Provide any output in the Tomcat log (typically catalina.out) related to your problem.
- Web browser console: In the Chrome web browser, click F12, then "Console", and look for exceptions related to your problem.
- Actions leading to the problem: Describe as clearly as possible which steps you take leading to the problem or exception.
- Problem description: Describe the problem clearly, why you think it is a problem and which behavior you would expect from the system.

11 Organisation Units

In DHIS2 the location of the data, the geographical context, is represented as organisational units. Organisational units can be either a health facility or department/sub-unit providing services or an administrative unit representing a geographical area (e.g. a health district).

Organisation units are located within a hierarchy, also referred to as a tree. The hierarchy will reflect the health administrative structure and its levels. Typical levels in such a hierarchy are the national, province, district and facility levels. In DHIS2 there is a single organisational hierarchy so the way this is defined and mapped to the reality needs careful consideration. Which geographical areas and levels are defined in the main organisational hierarchy will have major impact on the usability and performance of the application. Additionally, there are ways of addressing alternative hierarchies and levels as explained in the section called Organisation unit groups and group sets further down.

11.1 Organisation unit hierarchy design

The process of designing a sensible organisation unit hierarchy has many aspects:

- *Include all reporting health facilities:* All health facilities which contribute to the national data collection should be included in the system. Facilities of all kinds of ownership should be incorporated, including private, public, NGO and faith-oriented facilities. Often private facilities constitute half of the total number of facilities in a country and have policies for data reporting imposed on them, which means that incorporating data from such facilities are necessary to get realistic, national aggregate figures.

- *Emphasize the health administrative hierarchy:* A country typically has multiple administrative hierarchies which are often not well coordinated nor harmonized. When considering what to emphasize when designing the DHIS2 database one should keep in mind what areas are most interesting and will be most frequently requested for data analysis. DHIS2 is primarily managing health data and performing analysis based on the health administrative structure. This implies that even if adjustments might be made to cater for areas such as finance and local government, the point of departure for the DHIS2 organisation unit hierarchy should be the health administrative areas.

- *Limit the number of organisation unit hierarchy levels:* To cater for analysis requirements coming from various organisational bodies such as local government and the treasury, it is tempting to include all of these areas as organisation units in the DHIS2 database. However, due to performance considerations one should try to limit the organisation unit hierarchy levels to the smallest possible number. The hierarchy is used as the basis for aggregation of data to be presented in any of the reporting tools, so when producing aggregate data for the higher levels, the DHIS2 application must search for and add together data registered for all organisation units located further down the hierarchy. Increasing the number of organisation units will hence negatively impact the performance of the application and an excessively large number might become a significant problem in that regard.

  In addition, a central part in most of the analysis tools in DHIS2 is based around dynamically selecting the “parent” organisation unit of those which are intended to be included. For instance, one would want to select a province and have the districts belonging to that province included in the report. If the district level is the most interesting level from an analysis point of view and several hierarchy levels exist between this and the province level, this kind of report will be rendered unusable. When building up the hierarchy, one should focus on the levels that will be used frequently in reports and data analysis and leave out levels that are rarely or never used as this will have an impact on both the performance and usability of the application.

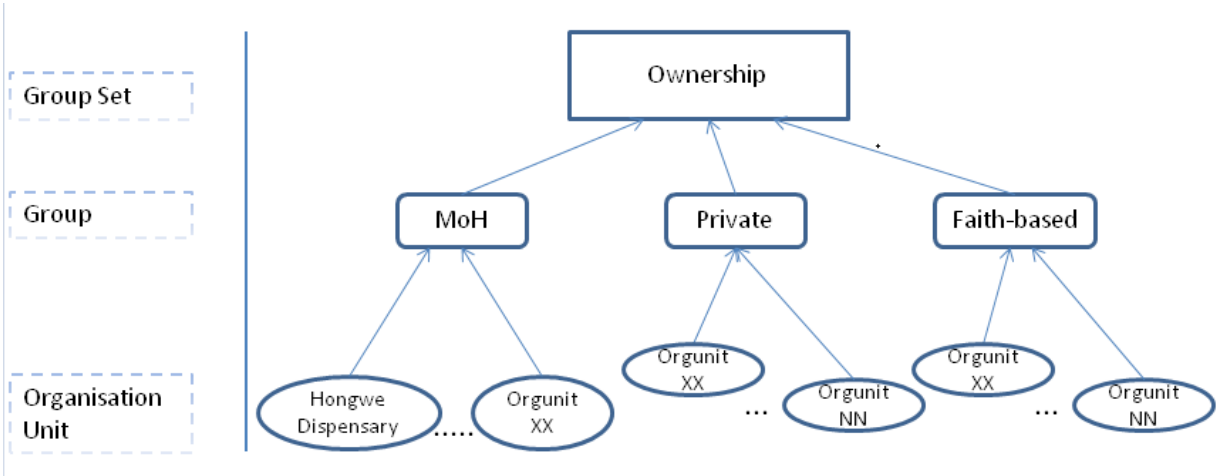
- *Avoid one-to-one relationships:* Another guiding principle for designing the hierarchy is to avoid connecting levels that have near one-to-one parent-child ratios, meaning that for instance a district (parent) should have on average more than one local council (child) at the level below before it make sense to add a local council level to the hierarchy. Parent-child ratios from 1:4 or more are much more useful for data analysis purposes as one can start to look at e.g. how a district's data is distributed in the different sub-areas and how these vary. Such drill-down exercises are not very useful when the level below has the same target population and the same serving health facilities as the parent.

  Skipping geographical levels when mapping the reality to the DHIS2 organisation unit hierarchy can be difficult and can easily lead to resistance among certain stakeholders, but one should have in mind that there are actually ways of producing reports based on geographical levels that are not part of the organisational hierarchy in DHIS2, as will be explained in the next section.

11.2 Organisation unit groups and group sets

In DHIS2, organisation units can be grouped in organisation unit groups, and these groups can be further organised into group sets. Together they can mimic an alternative organisational hierarchy which can be used when creating reports and other data output. In addition to representing alternative geographical locations not part of the main hierarchy, these groups are useful for assigning classification schemes to health facilities, e.g. based on the type or ownership of the facilities. Any number of group sets and groups can be defined in the application through the user interface, and all these are defined locally for each DHIS2 database.

An example illustrates this best: Typically one would want to provide analysis based on the ownership of the facilities. In that case one would create a group for each ownership type, for instance "MoH", "Private" and "NGO". All facilities in the database must then be classified and assigned to one and only one of these three groups. Next one would create a group set called "Ownership" to which the three groups above are assigned, as illustrated in the figure below.



In a similar way one can create a group set for an additional administrative level, e.g. local councils. All local councils must be defined as organisation unit groups and then assigned to a group set called "Local Council". The final step is then to assign all health facilities to one and only one of the local council groups. This enables the DHIS2 to produce aggregate reports by each local council (adding together data for all assigned health facilities) without having to include the local council level in the main organisational hierarchy. The same approach can be followed for any additional administrative or geographical level that is needed, with one group set per additional level. Before going ahead and designing this in DHIS2, a mapping between the areas of the additional geographical level and the health facilities in each area is needed.

A key property of the group set concept in DHIS2 to understand is *exclusivity*, which implies that an organisation unit can be member of exactly one of the groups in a group set. A violation of this rule would lead to duplication of data when aggregating health facility data by the different groups, as a facility assigned to two groups in the same group set would be counted twice.

With this structure in place, DHIS2 can provide aggregated data for each of the organisation unit ownership types through the “Organisation unit group set report” in “Reporting” module or through the Excel pivot table third-party tool. For instance one can view and compare utilisation rates aggregated by the different types of ownership (e.g. MoH, Private, NGO). In addition, DHIS2 can provide statistics of the distribution of facilities in “Organisation unit distribution report” in “Reporting” module. For instance one can view how many facilities exist under any given organisation unit in the hierarchy for each of the various ownership types. In the GIS module, given that health facility coordinates have been registered in the system, one can view the locations of the different types of health facilities (with different symbols for each type), and also combine this information with another map layer showing indicators e.g. by district.

12 Data Elements and Custom Dimensions

This chapter first discusses an important building block of the system: the data element. Second it discusses the category model and how it can be used to achieve highly customized meta-data structure for storage of data.

12.1 Data elements

The data element is together with the organisation unit the most important building block of a DHIS2 database. It represents the *what* dimension and explains what is being collected or analysed. In some contexts this is referred to an indicator, however in DHIS2 this meta-data element of data collection and analysis is referred to as a data element. The data element often represents a count of some event and its name describes what is being counted, e.g. “BCG doses given” or “Malaria cases”. When data is collected, validated, analysed or presented it is the data elements or expressions built with data elements that describe what phenomenon, event or case the data is registered for. Hence the data elements become important for all aspects of the system and decide not only how data is collected, but more importantly how the data is represented in the database and how data can be analysed and presented.

An important principle behind designing data elements is to think of data elements as a self-contained description of an phenomenon or event and not as a field in a data entry form. Each data element lives on its own in the database, completely detached and independent from the collection form. It is important to consider that data elements are used directly in reports, charts and other tools for data analysis, in which the context in any given data entry form is not accessible nor relevant. In other words, it must be possible to clearly identify what event a data element represents by only looking at its name. Based on this one can derive a rule of thumb saying that the name of the data element must be able to stand on its own and describe the data value also outside the context of its collection form.

For instance, a data element called “Malaria” might be concise when seen in a data entry form capturing immunization data, in a form capturing vaccination stocks as well as in a form for out-patient data. When viewed in a report, however, outside the context of the data entry form, it is impossible to decide what event this data element represents. If the data element had been called “Malaria cases”, “Malaria stock doses received” or “Malaria doses given” it would have been clear from a user perspective what the report is trying to express. In this case we are dealing with three different data elements with completely different semantics.

12.2 Categories and custom dimensions

Certain requirements for data capture necessitate a fine-grained breakdown of the dimension describing the event being counted. For instance one would want to collect the number of “Malaria cases” broken down on gender and age groups, such as “female”, “male” and “< 5 years” and “> 5 years”. What characterizes this is that the breakdown is typically repeated for a number of “base” data elements: For instance one would like to reuse this break-down for other data elements such as “TB” and “HIV”. In order to make the meta-data more dynamic, reusable and suitable for analysis it makes sense to define the mentioned diseases as data elements and create a separate model for the breakdown attributes. This can be achieved by using the category model, which is described in the following.

The category model has three main elements which is best described using the above example:

1. The category option, which corresponds to “female”, “male” and “< 5 years” and “> 5 years”.
2. The category, which corresponds to “gender” and “age group”.

3. The category combination, which should in the above example be named “gender and age group” and be assigned both categories mentioned above.

This category model is in fact self-standing but is in DHIS2 loosely coupled to the data element. Loosely coupled in this regard means that there is an association between data element and category combination, but this association may be changed at any time without loosing any data. It is however not recommended to change this often since it makes the database less valuable in general since it reduces the continuity of the data. Note that there is no hard limit on the number of category options in a category or number of categories in a category combination, however there is a natural limit to where the structure becomes messy and unwieldy.

A pair of data element and category combination can now be used to represent any level of breakdown. It is important to understand that what is actually happening is that a number of custom dimensions are assigned to the data. Just like the data element represents a mandatory dimension to the data values, the categories add custom dimensions to it. In the above example we can now, through the DHIS2 output tools, perform analysis based on both “gender” and “age group” for those data elements, in the same way as one can perform analysis based on data elements, organisation units and periods.

This category model can be utilized both in data entry form designs and in analysis and tabular reports. For analysis purposes, DHIS2 will automatically produce sub-totals and totals for each data element associated with a category combination. The rule for this calculation is that all category options should sum up to a meaningful total. The above example shows such a meaningful total since when summarizing “Malaria cases” captured for “female < 5 years”, “male < 5 years”, “female > 5 years” and “male > 5 years” one will get the total number of “Malaria cases”.

For data capture purposes, DHIS2 can automatically generate tabular data entry forms where the data elements are represented as rows and the category option combinations are represented as columns. This will in many situations lead to compelling forms with a minimal effort. It is necessary to note that this however represents a dilemma as these two concerns are sometimes not compatible. For instance one might want to quickly create data entry forms by using categories which do not adhere to the rule of a meaningful total. We do however consider this a better alternative than maintaining two independent and separate models for data entry and data analysis.

An important point about the category model is that data values are persisted and associated with a category option combination. This implies that adding or removing categories from a category combination renders these combinations invalid and a low-level database operation must be done to correct it. It is hence recommended to thoughtfully consider which breakdowns are required and to not change them too often.

12.3 Data element groups

Common properties of data elements can be modelled through what is called data element groups. The groups are completely flexible in the sense that both their names and their memberships are defined by the user. Groups are useful both for browsing and presenting related data, and can also be used to aggregate values captured for data elements in the group. Groups are loosely coupled to data elements and not tied directly to the data values which means they can be modified and added at any point in time without interfering with the low-level data.

13 Data Sets and Forms

This chapter discusses data sets and forms, what types of forms are available and describes best practises for the process of moving from paper based to electronic forms.

13.1 What is a data set?

All data entry in DHIS2 is organised through the use of data sets. A data set is a collection of data elements grouped together for data collection, and in the case of distributed installs they also define chunks of data for export and import between instances of DHIS2 (e.g. from a district office local installation to a national server). Data sets are not linked directly to the data values, only through their data elements and frequencies, and as such a data set can be modified, deleted or added at any point in time without affecting the raw data already captured in the system, but such changes will of course affect how new data will be collected.

A data set has a period type which controls the data collection frequency, which can be daily, weekly, monthly, quarterly, six-monthly, or yearly. Both the data elements to include in the data set and the period type is defined by the user, together with a name, short name, and code. If calculated fields are needed in the collection form (and not only in the reports), then indicators can be assigned to the data set as well, but these can only be used in custom forms (see further down).

In order to use a data set to collect data for a specific organisation unit the user must assign the organisation unit to the data set. This mechanism controls which organisation units can use which data sets, and at the same time defines the target values for data completeness (e.g. how many health facilities in a district are expected to submit the RCH data set every month).

A data element can belong to multiple data sets, but this requires careful thinking as it may lead to overlapping and inconstant data being collected if e.g. the data sets are given different frequencies and are used by the same organisation units.

13.2 What is a data entry form?

Once you have assigned a data set to an organisation unit that data set will be made available in Data Entry (under Services) for the organisation units you have assigned it to and for the valid periods according to the data set's period type. A default data entry form will then be shown, which is simply a list of the data elements belonging to the data set together with a column for inputting the values. If your data set contains data elements with categories such as age groups or gender, then additional columns will be automatically generated in the default form based on the categories. In addition to the default list-based data entry form there are two more alternatives, the section-based form and the custom form.

13.2.1 Types of data entry forms

DHIS2 currently features three differnet types of forms which are described in the following.

13.2.1.1 Default forms

A default data entry form is simply a list of the data elements belonging to the data set together with a column for inputting the values. If your data set contains data elements with a non-default category combination, such as age groups or gender then additional columns will be automatically generated in the default form based on the different options/dimensions. If you use more than one category combination in a data set you will get one table per category combination in the default form, with different column headings for the options.

13.2.1.2 Section forms

Section forms allow for a bit more flexibility when it comes to using tabular forms and are quick and simple to design. Often your data entry form will need multiple tables with subheadings, and sometimes you need to disable (grey out) a few fields in the table (e.g. some categories do not apply to all data elements), both of these functions are supported in section forms. After defining a data set you can define it's sections with subsets of data elements, a heading and possible grey fields in the section's table. The order of sections in a data set can also be defined. In Data Entry you can now start using the Section form (which should appear automatically when sections are available for the selected data set). Most tabular data entry forms should be possible to do with sections forms. Utilizing the section or default forms makes life easy as there is no need to maintain a fixed form design which includes references to data elements. If these two types of forms are not meeting your requirements then the third option is the completely flexible, although more time-consuming, custom data entry forms.

13.2.1.3 Custom Forms

When the form you want to design is too complicated for the default or section forms then your last option is to use a custom form. This takes more time, but gives you full flexibility in terms of the design. In DHIS2 there is a built in HTML editor (CK Editor) in the form designer which allows you to either design the form in the GUI or paste in your html directly (using the "source" window in the editor). In the custom form you can insert static text or data fields (linked to data elements + category option combination) in any position on the form and you have complete freedom to design the layout of the form. Once a custom form has been added to a data set it will be available in Data Entry and used automatically.

When using a custom form it is possible to use calculated fields to display e.g. running totals or other calculations based on the data captured in the form. This can e.g. be useful when dealing with stock or logistics forms that need item balance, items needed for next period etc. In order to do so, the user must first define the calculated expressions as indicators and then assign these indicators to the data set in question. In the custom form designer the user can then assign indicators to the form the same way data elements are assigned. The limitation to the calculated expression is that all the data elements used in the expression must be available in the same data set since the calculations are done on the fly inside the form, and are not based on data values already stored in the database.

13.3 From paper to electronic form - Lessons learned

When introducing an electronic health information system the system being replaced is often paper based reporting. The process of migrating to electronic data capture and analysis has some challenges. The following sections suggest best practises on how to overcome these.

13.3.1 Identify self-contained data elements

Typically the design of a DHIS2 data set is based on some requirements from a paper form that is already in use. The logic of paper forms are not the same as the data element and data set model of DHIS, e.g. often a field in a tabular paper form is described both by column headings and text on each row, and sometimes also with some introductory table heading that provides more context. In the database this is captured for one atomic data element with no reference to a position in a visual table format so it is important to make sure the data element, with the optional data element categories, captures the full meaning of each individual field in the paper form.

13.3.2 Leave calculations and repetitions to the computer - capture raw data only

Another important thing to have in mind while designing data sets is that the data set and the corresponding data entry form (which is a data set with layout) is a data collection tool and not a report or analysis tool. There are other far more sophisticated tools for data output and reporting in DHIS2 than the data entry forms. Paper forms are often designed with both data collection and reporting in mind and therefore you might see things such as cumulative values (in addition to the monthly values), repetition of annual data (the same population data reported every month) or even indicator values such as coverage rates in the same form as the monthly raw data. When you store the raw data in the DHIS2 database every month and have all the processing power you need within the computerised tool, there is no need (in fact it would be wrong and most likely cause inconsistency) to register manually calculated values such as the ones mentioned above. You only want to capture the raw data in your data sets/forms and leave the calculations to the computer, and presentation of such values to the reporting tools in DHIS. Through the functionality of data set reports all tabular section forms will automatically get extra columns at the far right providing subtotal and total values for each row (data element).

14 Data Quality

This chapter discusses various aspects related to data quality.

14.1 Measuring data quality

Is the data complete? Is it collected on time? Is it correct? These are questions that need to be asked when analysing data. Poor data quality can take many shapes; not just incorrect figures, but a lack of completeness, or the data being too old (for meaningful use).

14.2 Reasons for poor data quality

There are many potential reasons for poor quality data, including:

- Excessive amounts collected; too much data to be collected leads to less time to do it, and "shortcuts" to finish reporting
- Many manual steps; moving figures, summing up, etc. between different paper forms
- Unclear definitions; wrong interpretation of the fields to be filled out
- Lack of use of information: no incentive to improve quality
- Fragmentation of information systems; can lead to duplication of reporting

14.3 Improving data quality

Improving data quality is a long-term task, and many of the measures are organizational in nature. However, data quality should be an issue from the start of any implementation process, and there are some things that can be addressed at once, such as checks in DHIS2. Some important data quality improvement measures are:

- Changes in data collection forms, harmonization of forms
- Promote information use at local level, where data is collected
- Develop routines on checking data quality
- Include data quality in training
- Implement data quality checks in DHIS2

14.4 Using DHIS2 to improve data quality

DHIS2 has several features that can help the work of improving data quality; validation during data entry to make sure data is captured in the right format and within a reasonable range, user-defined validation rules based on mathematical relationships between the data being captured (e.g. subtotals vs totals), outlier analysis functions, as well as reports on data coverage and completeness. More indirectly, several of the DHIS2 design principles contribute to improving data quality, such as the idea of harmonising data into one integrated data warehouse, supporting local level access to data and analysis tools, and by offering a wide range of tools for data analysis and dissemination. With more structured and harmonised data collection processes and with strengthened information use at all levels, the quality of data will improve. Here is an overview of the functionality more directly targeting data quality:

14.4.1 Data input validation

The most basic type of data quality check in DHIS2 is to make sure that the data being captured is in the correct format. The DHIS2 will give the users a message that the value entered is not in the

correct format and will not save the value until it has been changed to an accepted value. E.g. text cannot be entered in a numeric field. The different types of data values supported in DHIS2 are explained in the user manual in the chapter on data elements.

14.4.2 Min and max ranges

To stop typing mistakes during data entry (e.g typing '1000' instead of '100') the DHIS2 checks that the value being entered is within a reasonable range. This range is based on the previously collected data by the same health facility for the same data element, and consists of a minimum and a maximum value. As soon as a the users enters a value outside the user will be alerted that the value is not accepted. In order to calculate the reasonable ranges the system needs at least six months (periods) of data.

14.4.3 Validation rules

A validation rule is based on an expression which defines a relationship between a number of data elements. The expression has a left side and a right side and an operator which defines whether the former must be less than, equal to or greater than the latter. The expression forms a condition which should assert that certain logical criteria are met. For instance, a validation rule could assert that the total number of vaccines given to infants is less than or equal to the total number of infants.

The validation rules can be defined through the user interface and later be run to check the existing data. When running validation rules the user can specify the organisation units and periods to check data for, as running a check on all existing data will take a long time and might not be relevant either. When the checks have completed a report will be presented to the user with validation violations explaining which data values need to be corrected.

The validation rules checks are also built into the data entry process so that when the user has completed a form the rules can be run to check the data in that form only, before closing the form.

14.4.4 Outlier analysis

The standard deviation based outlier analysis provides a mechanism for revealing values that are numerically distant from the rest of the data. Outliers can occur by chance, but they often indicate a measurement error or a heavy-tailed distribution (leading to very high numbers). In the former case one wishes to discard them while in the latter case one should be cautious in using tools or interpretations that assume a normal distribution. The analysis is based on the standard normal distribution.

14.4.5 Completeness and timeliness reports

Completeness reports will show how many data sets (forms) have been submitted by organisation unit and period. You can use one of three different methods to calculate completeness; 1) based on completeness button in data entry, 2) based on a set of defined compulsory data elements, or 3) based on the total registered data values for a data set.

The completeness reports will also show which organisation units in an area are reporting on time, and the percentage of timely reporting facilities in a given area. The timeliness calculation is based on a system setting called Days after period end to qualify for timely data submission.

15 Indicators

This chapter covers the following topics:

- What is an indicator
- Purposes of indicators
- Indicator-driven data collection
- Managing indicators in DHIS2

The following describes these topics in greater detail.

15.1 What is an indicator?

In DHIS2, the indicator is a core element of data analysis. An indicator is a calculated formula based on a combination of data elements, category options, possibly constants and a factor. There are two foms of indicators, those with a denominator and those which do not have a denominator. Calculated totals, which may be composed of multiple data elements do not have denominators. Coverage indicators (ratios, percentages, etc) are composed of two formulas of data elements, one representing the numerator and another representing the denominator.

Indicators are thus made up of formulas of data elements and other components and are always multiplied by a factor (e.g. 1, 100, 100, 100 000). The factor is essentially a number which is multiplied by the result of the numerator divided by denominator. As a concrete example, the indicator “BCG coverage <1 year” is defined by a formula with a factor 100 (in order to obtain a percentage), a numerator (“BCG doses given to children under 1 year”) and a denominator (“Target population under 1 year”). The indicator “DPT1 to DPT3 drop out rate” is a formula of 100 % x (“DPT1 doses given”- “DPT3 doses given”) / (“DPT1 doses given”).

Indicator examples

Indicator	Formula	Numerator	Denominator	Factor
Fully immunized <1 year coverage	Fully immunized/ Population < 1 year x 100	Fully immunized	Population < 1	100 (Percentage)
Maternal Mortality Rate	Maternal deaths/Live births x 100 000	Maternal deaths	Live births	100 000 (MMR is measured per 100 000)

Indicator	Formula	Numerator	Denominator	Factor
Cumulative number of people Enrolled in Care	Cumulative number of people Enrolled in Care x 1	Cumulative number Enrolled in Care (Male, Age<18) +Cumulative number Enrolled in Care (Male, Age18+) +Cumulative number Enrolled in Care (Female, Age<18) +Cumulative number Enrolled in Care (Female, Age18+)	None	1

15.2 Purpose of indicators

Indicators which are defined with both numerators and denominators are typically more useful for analysis. Because they are proportions, they are comparable across time and space, which is very important since units of analysis and comparison, such as districts, vary in size and change over time. A district with population of 1000 people may have fewer cases of a given disease than a district with a population of 10,000. However, the incidence values of a given disease will be comparable between the two districts because of the use of the respective populations for each district.

Indicators thus allow comparison, and are the prime tool for data analysis. DHIS2 should provide relevant indicators for analysis for all health programs, not just the raw data. Most report modules in DHIS2 support both data elements and indicators and you can also combine these in custom reports.

15.3 Indicator-driven data collection

Since indicators are more suited for analysis compared to data elements, the calculation of indicators should be the main driving force for collection of data. A usual situation is that much data is collected but never used in any indicator, which significantly reduces the usability of the data. Either the captured data elements should be included in indicators used for management or they should probably not be collected at all.

For implementation purposes, a list of indicators used for management should be defined and implemented in DHIS2. For analysis, training should focus on the use of indicators and why these are better suited than data elements for this purpose.

15.4 Managing indicators

Indicators can be added, deleted, or modified at any time in DHIS2 without affecting the data. Indicators are not stored as values in DHIS2, but as formulas, which are calculated whenever the user needs them. Thus a change in the formulas will only lead to different data elements being called for when using the indicator for analysis, without any changes to the underlying data values taking place. For information how to manage indicators, please refer to the chapter on indicators in the DHIS2 user documentation.

16 Users and user roles

16.1 About user management

Multiple users can access DHIS2 simultaneously and each user can have different authorities. You can fine-tune these authorities so that certain users can only enter data, while others can only generate reports.

- You can create as many users, user roles and user groups as you need.
- You can assign specific authorities to user groups or individual users via user roles.
- You can create multiple user roles each with their own authorities.
- You can assign user roles to users to grant the users the corresponding authorities.
- You can assign each user to organisation units. Then the user can enter data for the assigned organisation units.

User management terms and definitions

Term	Definition	Example
Authority	A permission to perform one or several specific tasks	Create a new data element Update an organisation unit View a report
User	A person's DHIS2 user account	admin traore guest
User role	A group of authorities	Data entry clerk System administrator Antenatal care program access
User group	A group of users	Kenya staff Feedback message recipients HIV program coordinators

You manager users, user roles and user groups in the **Users** app.

Objects in the Users app

Object type	Available functions
User	Create, edit, invite, clone, disable, display by organisation unit, delete and show details
User role	Create, edit, share, delete and show details
User group	Create, edit, join, leave, share, delete and show details

16.1.1 About users

Each user in DHIS2 must have a user account which is identified by a user name. You should register a first and last name for each user as well as contact information, for example an email address and a phone number.

It is important that you register the correct contact information. DHIS2 uses this information to contact users directly, for example sending emails to notify users about important events. You can also use the contact information to share for example dashboards and pivot tables.

A user in DHIS2 is associated with an organisation unit. You should assign the organisation unit where the user works.

When you create a user account for a district record officer, you should assign the district where he/she works as the organisation unit.

The assigned organisation unit affects how the user can use DHIS2:

- In the **Data Entry** app, a user can only enter data for the organisation unit she is associated with and the organisation units below that in the hierarchy. For instance, a district records officer will be able to register data for her district and the facilities below that district only.
- In the **Users** app, a user can only create new users for the organisation unit she is associated with in addition to the organisation units below that in the hierarchy.
- In the **Reports** app, a user can only view reports for her organisation unit and those below. (This is something we consider to open up to allow for comparison reports.)

An important part of user management is to control which users are allowed to create new users with which authorities. In DHIS2, you can control which users are allowed to perform this task. The key principle is that a user can only grant authorities and access to data sets that the user itself has access to. The number of users at national, province and district level are often relatively few and can be created and managed by the system administrators. If a large proportion of the facilities are entering data directly into the system, the number of users might become unwieldy. It is recommended to delegate and decentralize this task to the district officers, it will make the process more efficient and support the facility users better.

16.1.2 About user roles

A user role in DHIS2 is a group of authorities. An authority means the permission to perform one or more specific tasks.

A user role can contain authorities to create a new data element, update an organisation unit or view a report.

A user can have multiple user roles. If so, the user's authorities will be the sum of all authorities and data sets in the user roles. This means that you can mix and match user roles for special purposes instead of only creating new ones.

A user role is associated with a collection of data sets. This affects the **Data Entry** app: a user can only enter data for the data sets registered for his/her user role. This can be useful when, for example, you want to allow officers from health programs to enter data only for their relevant data entry forms.

Recommendations:

- Create one user role for each position within the organisation.
- Create the user roles in parallel with defining which user is doing which tasks in the system.
- Only give the user roles the exact authorities they need to perform their job, not more. Only those who are supposed to perform a task should have the authorities to perform it.

16.1.3 About user groups

A user group is a group of users. You use user groups when you set up sharing of objects or notifications, for example push reports or program notifications.

See also:

Sharing

Manage program notifications

Mange push reports

16.2 Workflow

1. Define the positions you need for your project and identify which tasks the different positions will perform.
2. Create roughly one user role for each position.
3. Create users.
4. Assign user roles to the users.
5. Assign the users to organisation units.
6. (Optional) Group users in user groups.
7. Share datasets with users or user-groups via the Sharing Dialog in Data set management section of the Maintenance app

> **Tip**
>
> For users to be able to enter data, you must add them to an organisational unit level and share a dataset with them.

16.3 Example: user management in a health system

In a health system, users are logically grouped with respect to the task they perform and the position they occupy.

1. Define which users should have the role as system administrators. They are often part of the national HIS division and should have full authority in the system.
2. Create roughly one user role for each position.

Examples of common positions are:

Position	Typical tasks	Recommended authorities	Comment
System administrators	Set up the basic structure (metadata) of the system.	Add, update and delete the core elements of the system, for example data elements, indicators and data sets.	Only system administrators should modify metadata. If you allow users outside the system administrators team to modify the metadata, it might lead to problems with coordination. Updates to the system should only be performed by the administrators of the system.
National health managers Province health managers	Monitor and analyse data	Access to the reports module, the **GIS**, **Data Quality** apps and the dashboard.	Don't need access to enter data, modify data elements or data sets.
National health information system division officers (HISO) District health records and information officers (DHRIO) Facility health records and information officers (HRIO)	Enter data that comes from facilities which are not able to do so directly Monitor, evaluate and analyse data	Access to all the analysis and validation apps Access to the **Data Entry** app.	-

Position	Typical tasks	Recommended authorities	Comment
Data entry clerks	-	-	-

17 Data Analysis Tools Overview

This chapter offers an overview of the available tools for data analysis provided by DHIS2 along with a description of the purpose and benefits of each. If you are looking for a detailed guide on how to use each tool we recommend to continue to read the user guide after finishing this chapter. The following list shows the various tools:

1. Standard reports
2. Data set reports
3. Data completeness reports
4. Static reports
5. Organisation unit distribution reports
6. Report tables
7. Charts
8. Web Pivot table
9. GIS
10. My Datamart and Excel pivot tables

17.1 Data analysis tools

The following section gives a description of each tool.

17.1.1 Standard reports

Standard reports are reports with predefined designs. This means that the reports are easily accessible with a few clicks and can be consumed by users at all levels of experience. The report can contain statistics in the form of tables and charts and can be tailored to suit most requirements. The report solution in DHIS2 is based on JasperReports and reports are most often designed with the iReport report designer. Even though the report design is fixed, data can be dynamically loaded into the report based on any organisation unit from within the hierarchy and with a variety of time periods.

17.1.2 Data set reports

Data set reports displays the design of data entry forms as a report populated with aggregated data (as opposed to captured low-level data). This report is easily accessible for all types of users and gives quick access to aggregate data. There is often a legacy requirement for viewing data entry forms as reports which this tool efficiently provides for. The data set report supports all types of data entry forms including section and custom forms.

17.1.3 Data completeness report

The data completeness report produces statistics for the degree of completeness of data entry forms. The statistical data can be analysed per individual data sets or per a list of organisation units with a common parent in the hierarchy. It provides a percentage value for the total completeness and for the completeness of timely submissions. One can use various definitions of completeness as basis for the statistics: First based on number of data sets marked manually as complete by the user entering data. Second based on whether all data element defined as compulsory are being filled in for a data set. Third based on the percentage of number of values filled over the total number of values in a data set.

17.1.4 Static reports

Static reports provides two methods for linking to existing resources in the user interface. First it provides the possibility to link to a resource on the Internet trough a URL. Second it provides the possibility to upload files to the system and link to those files. The type of files to upload can be any kind of document, image or video. Useful examples of documents to link to are health surveys, policy documents and annual plans. URLs can point to relevant web sites such as the Ministry of Health home page, sources of health related information. In addition it can be used as an interface to third-party web based analysis tools by pointing at specific resources. One example is pointing a URL to a report served by the BIRT reporting framework.

17.1.5 Organisation unit distribution reports

The organisation unit distribution report provides statistics on the facilities (organisation units) in the hierarchy based on their classification. The classification is based on organisation unit groups and group sets. For instance facilities can be classified by type through assignment to the relevant group from the group set for organisation unit type. The distribution report produces the number of facilities for each class and can be generated for all organisation units and for all group sets in the system.

17.1.6 Report tables

Report tables are reports based on aggregated data in a tabular format. A report table can be used as a stand-alone report or can be used as data source for a more sophisticated standard report design. The tabular format can be cross-tabulated with any number of dimensions appearing as columns. It can contain indicator and data element aggregate data as well as completeness data for data sets. It can contain relative periods which enables the report to be reused over time. It can contain user selectable parameters for organisation units and periods to enable the report to be reused for all organisation units in the hierarchy. The report table can be limited to the top results and sorted ascending or descending. When generated the report table data can be downloaded as PDF, Excel workbook, CSV file and Jasper report.

17.1.7 Charts

The chart component offers a wide variety of charts including the standard bar, line and pie charts. The charts can contain indicators, data elements, periods and organisation units on both the x and y axis as well as a fixed horizontal target line. Charts can be view directly or as part of the dashboard as will be explained later.

17.1.8 Web Pivot tables

The web pivot table offers quick access to statistical data in a tabular format and provides the ability to “pivot” any number of the dimensions such as indicators, data elements, organisation units and periods to appear on columns and rows in order to create tailored views. Each cell in the table can be visualized as a bar chart.

17.1.9 GIS

The GIS module gives the ability to visualize aggregate data on maps. The GIS module can provide thematic mapping of polygons such as provinces and districts and of points such as facilities in separate layers. The mentioned layers can be displayed together and be combined with custom overlays. Such map views can be easily navigated back in history, saved for easy access at a later stage and saved to disk as an image file. The GIS module provides automatic and fixed class breaks for thematic mapping, predefined and automatic legend sets, ability to display labels (names) for the geographical elements and the ability to measure the distance between points in the map. Mapping can be viewed for any indicator or data element and for any level in

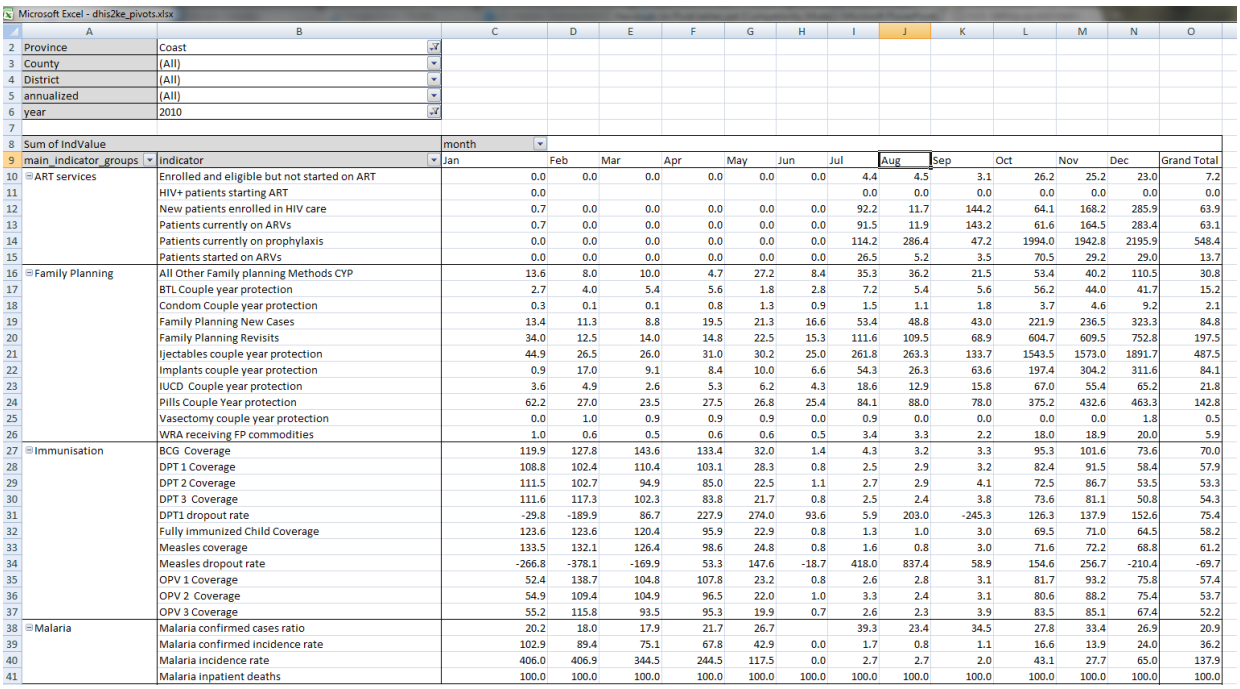
the organisation unit hierarchy. There is also a special layer for displaying facilities on the map where each one is represented with a symbol based on its type.

17.1.10 My Datamart and Excel Pivot tables

The purpose of the My Datamart tool is to provide users with full access to aggregate data even on unreliable Internet connections. This tool consists of a light-weight client application which is installed on the computer of the users. It connects to an online central server running a DHIS2 instance, downloads aggregate data and stores it in a database on the local computer. This database can be used to connect third-party tools such as MS Excel Pivot tables, which is a powerful tool for data analysis and visualization. This solution implies that just short periods of Internet connectivity are required to synchronize the client database with the central online one, and that after this process is done the data will be available independent of connectivity. Please read the chapter dedicated to this tool for in-depth information.

18 Pivot Tables and the MyDataMart tool

Excel Pivot Table (see screenshot below) is a powerful and dynamic data analysis tool that can be automatically linked to the DHIS2 data. While most reporting tools in DHIS2 are limited in how much data they can present at the same time, the pivot tables are designed to give nice overviews with multiple data elements or indicators, and organisation units and periods (see example below). Furthermore, the dynamic features in pivoting and drill-down are very different from static spreadsheets or many web reports, and this makes it a useful tool for information users who want to do more in-depth analysis and to manipulate the views on the data more dynamically. This, combined with the well-known charting capabilities of Excel, has made the Pivot Table tool a popular analysis tool among the more advanced DHIS2 users for a long time.



Microsoft Excel - dhis2ke_pivots.xlsx

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
2	Province	Coast													
3	County	(All)													
4	District	(All)													
5	annualized	(All)													
6	year	2010													
7															
8	Sum of IndValue		month												
9	main_indicator_groups	indicator	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Grand Total
10	ART services	Enrolled and eligible but not started on ART	0.0	0.0	0.0	0.0	0.0	0.0	4.4	4.5	3.1	26.2	25.2	23.0	7.2
11		HIV+ patients starting ART	0.0						0.0	0.0	0.0	0.0	0.0	0.0	0.0
12		New patients enrolled in HIV care	0.7	0.0	0.0	0.0	0.0	0.0	92.2	11.7	144.2	64.1	168.2	285.9	63.9
13		Patients currently on ARVs	0.7	0.0	0.0	0.0	0.0	0.0	91.5	11.9	143.2	61.6	164.5	283.4	63.1
14		Patients currently on prophylaxis	0.0	0.0	0.0	0.0	0.0	0.0	114.2	286.4	47.2	1994.0	1942.8	2195.9	548.4
15		Patients started on ARVs	0.0	0.0	0.0	0.0	0.0	0.0	26.5	5.2	3.5	70.5	29.2	29.0	13.7
16	Family Planning	All Other Family planning Methods CYP	13.6	8.0	10.0	4.7	27.2	8.4	35.3	36.2	21.5	53.4	40.2	110.5	30.8
17		BTL Couple year protection	2.7	4.0	5.4	5.6	1.8	2.8	7.2	5.4	5.6	56.2	44.0	41.7	15.2
18		Condom Couple year protection	0.3	0.1	0.1	0.8	1.3	0.9	1.5	1.1	1.8	3.7	4.6	9.2	2.1
19		Family Planning New Cases	13.4	11.3	8.8	19.5	21.3	16.6	53.4	48.8	43.0	221.9	236.5	323.3	84.8
20		Family Planning Revisits	34.0	12.5	14.0	14.8	22.5	15.3	111.6	109.5	68.9	604.7	609.5	752.8	197.5
21		Ijectables couple year protection	44.9	26.5	26.0	31.0	30.2	25.0	261.8	263.3	133.7	1543.5	1573.0	1891.7	487.5
22		Implants couple year protection	0.9	17.0	9.1	8.4	10.0	6.6	54.3	26.3	63.6	197.4	304.2	311.6	84.1
23		IUCD Couple year protection	3.6	4.9	2.6	5.3	6.2	4.3	18.6	12.9	15.8	67.0	55.4	65.2	21.8
24		Pills Couple Year protection	62.2	27.0	23.5	27.5	26.8	25.4	84.1	88.0	78.0	375.2	432.6	463.3	142.8
25		Vasectomy couple year protection	0.0	1.0	0.9	0.9	0.9	0.0	0.9	0.0	0.0	0.0	0.0	1.8	0.5
26		WRA receiving FP commodities	1.0	0.6	0.5	0.6	0.6	0.5	3.4	3.3	2.2	18.0	18.9	20.0	5.9
27	Immunisation	BCG Coverage	119.9	127.8	143.6	133.4	32.0	1.4	4.3	3.2	3.3	95.3	101.6	73.6	70.0
28		DPT 1 Coverage	108.8	102.4	110.4	103.1	28.3	0.8	2.5	2.9	3.2	82.4	91.5	58.4	57.9
29		DPT 2 Coverage	111.5	102.7	94.9	85.0	22.5	1.1	2.7	2.9	4.1	72.5	86.7	53.5	53.3
30		DPT 3 Coverage	111.6	117.3	102.3	83.8	21.7	0.8	2.5	2.4	3.8	73.6	81.1	50.8	54.3
31		DPT1 dropout rate	-29.8	-189.9	86.7	227.9	274.0	93.6	5.9	203.0	-245.3	126.3	137.9	152.6	75.4
32		Fully immunized Child Coverage	123.6	123.6	120.4	95.9	22.9	0.8	1.3	1.0	3.0	69.5	71.0	64.5	58.2
33		Measles coverage	133.5	132.1	126.4	98.6	24.8	0.8	1.6	0.8	3.0	71.6	72.2	68.8	61.2
34		Measles dropout rate	-266.8	-378.1	-169.9	53.3	147.6	-18.7	418.0	837.4	58.9	154.6	256.7	-210.4	-69.7
35		OPV 1 Coverage	52.4	138.7	104.8	107.8	23.2	0.8	2.6	2.8	3.1	81.7	93.2	75.8	57.4
36		OPV 2 Coverage	54.9	109.4	104.9	96.5	22.0	1.0	3.3	2.4	3.1	80.6	88.2	75.4	53.7
37		OPV 3 Coverage	55.2	115.8	93.5	95.3	19.9	0.7	2.6	2.3	3.9	83.5	85.1	67.4	52.2
38	Malaria	Malaria confirmed cases ratio	20.2	18.0	17.9	21.7	26.7		39.3	23.4	34.5	27.8	33.4	26.9	20.9
39		Malaria confirmed incidence rate	102.9	89.4	75.1	67.8	42.9	0.0	1.7	0.8	1.1	16.6	13.9	24.0	36.2
40		Malaria incidence rate	406.0	406.9	344.5	244.5	117.5	0.0	2.7	2.7	2.0	43.1	27.7	65.0	137.9
41		Malaria inpatient deaths	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

With the recent shift towards online deployments, the offline pivot tables in Excel also provide a useful alternative to the online reporting tools as they allow for local data analysis without Internet connectivity, which can be an advantage on unstable or expensive connections. Internet is only needed to download new data from the online server and, as soon as the data exists locally, working with the pivot tables require no connectivity. The MyDatamart tool, which is explained in detail further down, helps the users to maintain a local data mart file (small database) which is updated over the Internet against the online server, and then used as an offline data source that feeds the pivot tables with data.

18.1 Pivot table design

Typically an Excel pivot table file set up for DHIS2 will contain multiple worksheets with one pivot table on each sheet. A table can consist of either raw data values (by data elements) or indicator values, and will usually be named based on which level of the organisation unit hierarchy the source data is aggregated by as well as the period type (frequency e.g. Monthly, Yearly) of the data. A standard DHIS2 pivot table file includes the following pivot tables: District Indicators, District Data Monthly, District Data Yearly, Facility Indicators, Facility Data Monthly, Facility Data Yearly. In addition there might be more specialized tables that focus on specific programs and/or other period types.

One popular feature of pivot tables is to be able to drag-and-drop the various fields between the three positions page/filter, row, and columns, and thereby completely change the data view. These fields can be seen as dimensions to the data values and represent the dimensions in the DHIS2 data model; organisation unit (one field per level), data elements or indicators, periods,

and then a dynamically extended lists of additional dimensions representing organisation unit/ indicator/data element group sets and data element categories (see other chapters of this guide for details). In fact a dynamic pivot table is an excellent tool to represent the many dimensions created in the DHIS2, and makes it very easy to zoom in or out on each dimension, e.g. look at raw data values by individual age groups or just by its total, or in combination with other dimensions like gender. All the dimensions created in the DHIS2 will be reflected in the available fields list of each pivot table, and then it is up to the user to select which ones to use.

It is important to understand that the values in the pivot tables are non-editable and all the names and numbers are fetched directly from the DHIS2 database, which makes it different from a normal spreadsheet. In order to be edited, the contents of a pivot table must be copied to a normal spreadsheet, but this is rarely needed as all the names can be edited in DHIS2 (and then be reflected in the pivot tables on the next update). The names (captions) on each field however are editable, but not their contents (values).

18.2 Connecting to the DHIS2 database

Each pivot table has a connection to the DHIS2 database and makes use of a pivot source view (SQL query) in the database to fetch the data. These queries pull all their data from the data mart tables, so it is important to keep the data mart updated at all times in order to get the most recent data into the pivot tables. A pivot table can connect to a database on the local computer or on a remote server. This makes it well suitable for use in a local network where there is only one shared database and multiple client computers using pivot tables. Excel can also connect to databases running on Linux. The database connection used in the pivot tables is specified in an ODBC data source on the Windows computers running pivot tables.

For online deployments the recommended way to connect to the DHIS2 data is to make use of the MyDatamart tool, which creates and updates a local data mart file (database) that Excel can connect to. The MyDatamart tool will be described in detail further down.

18.3 Dealing with large amounts of data

The amount of data in a DHIS2 database can easily go beyond the capabilities of Excel. A table with around 1 million values (rows of data) tend to become less responsive to updates (refresh) and pivoting operations, and on some computers Excel will give out of memory errors when dealing with tables of this size. Typically, the more powerful the computer, the more data can be handled, but the top limit seems to be around 1 million rows even on the high-end computers.

To deal with this problem the standard DHIS2 pivot table setup is to split the data over several pivot tables. There are different ways of splitting the data; by organisation unit aggregation level (how deep), by organisation unit coverage/boundary area (how wide), by period (e.g. one year of data at a time), or by data element or indicator groups (e.g. by health programs or themes). Aggregating away the lowest level in the organisation unit hierarchy is the most effective approach as it reduces the amount of data by a factor of the number of health facilities in a country. Typically there is no need to look at all the health facilities in a country at the same time, but instead only for a limited area (e.g. district or province). And when there is a need for data for the whole country that can be provided with district level aggregates or similar. At a district or province office the users will typically have facility level data only for their own area, and then for the neighboring areas the data will be aggregated up one or two levels to reduce the size of data, but still allow for comparison, split into e.g. the two tables Facility Data and District Data, and similar for indicator values. Splitting data by period or by data element/indicator groups work more or less in the same way, and can be done either in combination with the organisation unit splitting or instead of it. E.g. if a health program wants to analyse a few data elements at facility level for the whole country that can be possible. The splitting is controlled by the pivot views in the database where one specifies which data values to fetch.

19 DHIS2 as a platform

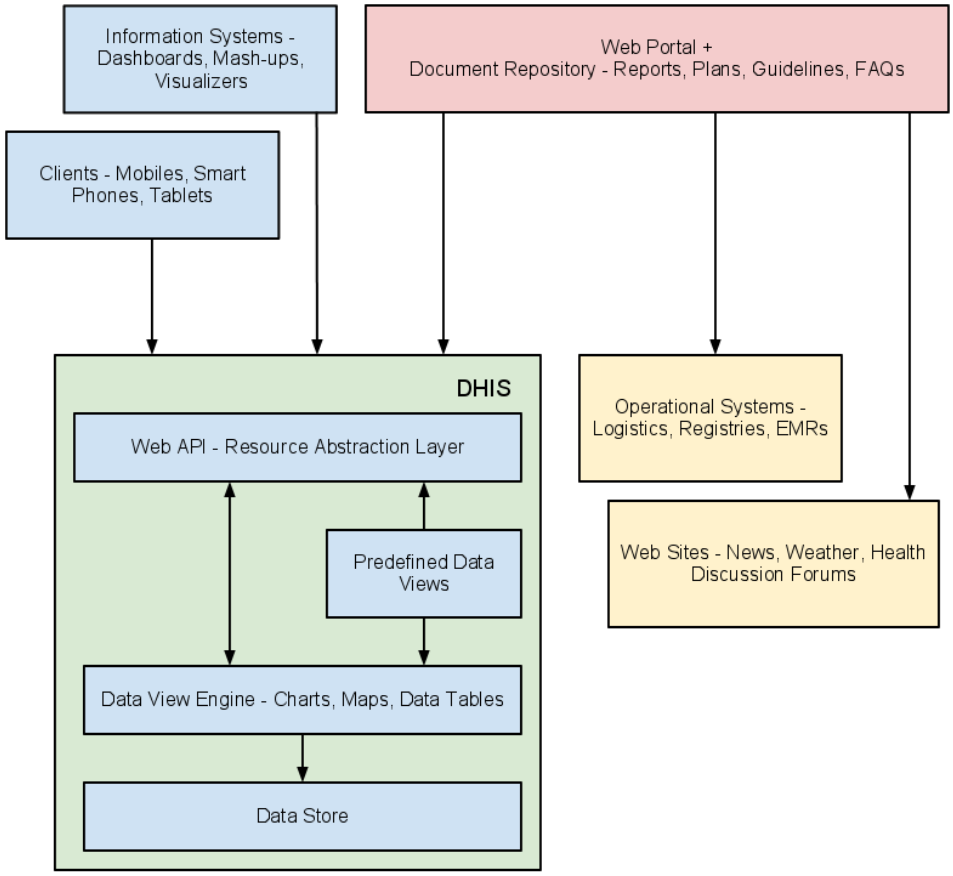
DHIS2 can be perceived as a platform on several levels. First, the application database is designed ground-up with flexibility in mind. Data structures such as data elements, organisation units, forms and user roles can be defined completely freely through the application user interface. This makes it possible for the system to be adapted to a multitude of local contexts and use-cases. We have seen that DHIS2 supports most major requirements for routine data capture and analysis emerging in country implementations. It also makes it possible for DHIS2 to serve as a management system for domains such as logistics, labs and finance.

Second, due to the modular design of DHIS2 it can be extended with additional software modules. These software modules can live side by side with the core modules of DHIS2 and can be integrated into the DHIS2 portal and menu system. This is a powerful feature as it makes it possible to extend the system with extra functionality when needed, typically for country specific requirements as earlier pointed out.

The downside of the software module extensibility is that it puts several constraints on the development process. The developers creating the extra functionality are limited to the DHIS2 technology in terms of programming language and software frameworks, in addition to the constraints put on the design of modules by the DHIS2 portal solution. Also, these modules must be included in the DHIS2 software when the software is built and deployed on the web server, not dynamically during run-time.

In order to overcome these limitations and achieve a looser coupling between the DHIS2 service layer and additional software artifacts, the DHIS2 development team decided to create a Web API. This Web API complies with the rules of the REST architectural style. This implies that:

- The Web API provides a navigable and machine-readable interface to the complete DHIS2 data model. For instance, one can access the full list of data elements, then navigate using the provided hyperlink to a particular data element of interest, then navigate using the provided hyperlink to the list of forms which this data element is part of. E.g. clients will only do state transitions using the hyperlinks which are dynamically embedded in the responses.
- Data is accessed through a uniform interface (URLs) using a well-known protocol. There are no fancy transport formats or protocols involved - just the well-tested, well-understood HTTP protocol which is the main building block of the Web today. This implies that third-party developers can develop software using the DHIS2 data model and data without knowing the DHIS2 specific technology or complying with the DHIS2 design constraints.
- All data including meta-data, reports, maps and charts, known as resources in REST terminology, can be retrieved in most of the popular representation formats of the Web of today, such as HTML, XML, JSON, PDF and PNG. These formats are widely supported in applications and programming languages and give third-party developers a wide range of implementation options.



There are several scenarios where additional software artifacts may connect to the DHIS2 Web API.

19.1 Web portals

First, Web portals may be built on top of the Web API. A Web portal in this regard is a web site which functions as a point of access to information from a potential large number of data sources which typically share a common theme. The role of the Web portal is to make such data sources easily accessible in a structured fashion under a common look-and-feel and provide a comprehensive data view for end users.

Aggregate data repository: A Web portal targeted at the health domain may use the DHIS2 as the main source for aggregate data. The portal can connect to the Web API and communicate with relevant resources such as maps, charts, reports, tables and static documents. These data views can dynamically visualize aggregate data based on queries on the organisation unit, indicator or period dimension. The portal can add value to the information accessibility in several ways. It can be structured in a user-friendly way and make data accessible to inexperienced users. It can provide various approaches to the data, including:

- Thematic - grouping indicators by topic. Examples of such topics are immunization, mother care, notifiable diseases and environmental health.
- Geographical - grouping data by provinces. This will enable easy comparison of performance and workload.

Mash-up: The Web portal is not limited to consuming data from a single Web API - it can be connected to any number of APIs and be used to mash up data from auxiliary systems within the health domain. If available the portal might pull in specialized data from logistics systems tracking and managing ARV medicines, from finance systems managing payments to health facilities and from lab systems tracking lab tests for communicable diseases. Data from all of

these sources might be presented in a coherent and meaningful way to provide better insight in the situation of the health domain.

Document repository: The Web portal can act as a document repository in itself (also referred to as content management system). Relevant documents such as published reports, survey data, annual operational plans and FAQs might be uploaded and managed in terms of ownership, version control and classification. This makes the portal a central point for document sharing and collaboration. The emergence of high-quality, open source repository/CMS solutions such as Alfresco and Drupal makes this approach more feasible and compelling.

Knowledge management: KM refers to practices for identifying, materializing and distributing insight and experience. In our context it relates to all aspects of information system implementation and use, such as:

- Database design
- Information system usage and how-to
- End-user training guidelines
- Data use, analysis and interpretation

Knowledge and learning within these areas can be materialized in the form of manuals, papers, books, slide sets, videos, system embedded help text, online learning sites, forums, FAQs and more. All of these artifacts might be published and made accessible from the Web portal.

Forum: The portal can provide a forum for hosting discussions between professional users. The subject can range from help for performing basic operations in the health information system to discussions over data analysis and interpretation topics. Such a forum can act as an interactive source for information and evolve naturally into a valuable archive.

19.2 Apps

Second, third-party software clients running on devices such as mobile phones, smart phones and tablets may connect to the DHIS2 Web API and read and write to relevant resources. For instance, third-party developers may create a client running on the Android operating system on mobile devices targeted at community health workers who needs to keep track of the people to visit, register vital data for each encounter and receive reminders of due dates for patient care while travelling freely in the community. Such a client application might interact with the patient and activity plan resources exposed by the DHIS2 Web API. The developer will not be dependent on deep insight into the DHIS2 internal implementation, rather just basic skills within HTTP/Web programming and a bit of knowledge of the DHIS2 data model. Understanding the DHIS2 data model is made easier by the navigable nature of the Web API.

19.3 Information Systems

Third, information system developers aiming at creating new ways of visualizing and presenting aggregate data can utilize the DHIS2 Web API as the service layer of their system. The effort needed for developing new information systems and maintaining them over time is often largely under-estimated. Instead of starting from scratch, a new application can be built on top of the Web API. Developer attention can be directed towards making new, innovative and creative data representations and visualizations, in the form of e.g. dashboards, GIS and charting components.

20 Localization of DHIS 2

20.1 DHIS 2 localization concepts

Localization involves the adaptation of an application to a specific location. When implementing DHIS 2 in a given country, adequate resources should be allocated to translate and localize the application if required. Translation of the user interface elements, messages, layout, date and time formats, currency and other aspects must be considered. In addition to translation of the user interface itself, metadata content which is contained in the database must also be considered to be translated.

DHIS 2 supports internationalization (i18n) of the user interface through the use of Java property strings and PO files. Java property files are used when messages originate from the back-end Java server, while PO files are used for front-end apps written in JavaScript. The DHIS 2 Android apps use a specific XML format. Regardless of the specific format, each element in the user interface has been assigned a specific key which is linked to a value. As an example,consider the following key/value pairs from a Java property file.

```
org_unit_tree=Organisation Unit Tree
error_occurred=An error has occurred.
```

In French the same key/value pairs would appear as follows

```
org_unit_tree=Arborescence des unités d'organisation
error_occurred=Une erreur s'est produite
```

Note that the keys (text before the = symbol) are the same in both examples, but the values (after the =) symbol are in each of the respective languages. Each of these values would need to be translated from the original language (English) to the destination language (e.g. French). When the user specifies French for the user interface language, all of the strings would then appear in French instead of the default language (English). Any strings which have not been translated, would appear in English.

There should always be an English string for all messages in DHIS 2. When the user selects a given language, and a translation is present in that language, then the translation will be shown. However, if the string in the desired language is missing then fallback rules will be applied. In cases when two given translations, such as Portuguese and Brazilian Portuguese share common messages, it is not required to perform a full translation in the variant language. Only messages which are different should be translated.
Fallback rules are then applied in the following manner (assuming the user has chooses Brazilian Portuguese as their language:

1. Display the message in Brazilian Portuguese if it exists.
2. If it does not exist in the variant language, then use the Portuguese message, if it exists.
3. If there is no message in either the base language or the variant language, choose the ultimate fallback language, English.

> **Important**
>
> - There are a number of key/value pairs such as “format.FinancialApril.startDate=dd MMM yyyy ‘to’” which are used

for date/time formatting in various parts of DHIS 2. Part of the value should not be translated because it is actually a special formatting field used by either Java or JavaScript to interpolate or format a string. In this example the special string which should be be translated is “dd MMM yyyy”. The part of the value which can be translated would be “to”, for instance to “a” in Spanish. If these data format template strings are translated, it may result in errors in the application.

- It is not necessary to translate a string from the original language (English) if the translated string is the same. You can simply leave it blank. By default, DHIS 2 will use English values for all strings which have not been translated.
- All translated strings must be stored in escaped UTF-8 format. If you are using the DHIS 2 translation portal (discussed below), be sure your browser settings are set to UTF-8 when translating. If you are using a text editor or other tool such as an IDE, you may need to convert the UTF-8 characters to escaped syntax, using the Java `native2ascii` utility.
- Some special variables (e.g. {0}) use curly brackets. This denotes a variable which will be replaced by a number or other value by the application. You must place this variable notation in the correct position and be sure not to modify it.

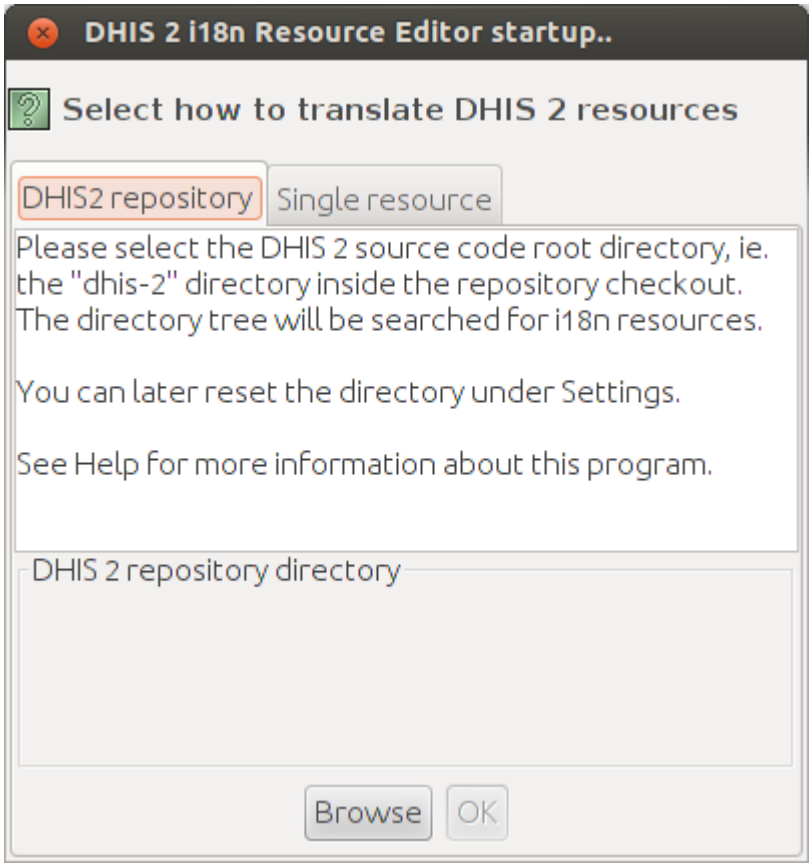
There are a number of tools which can be used to support the localization of the user interface as well as the database content, which will be discussed below.

20.2 DHIS 2 i18n tool

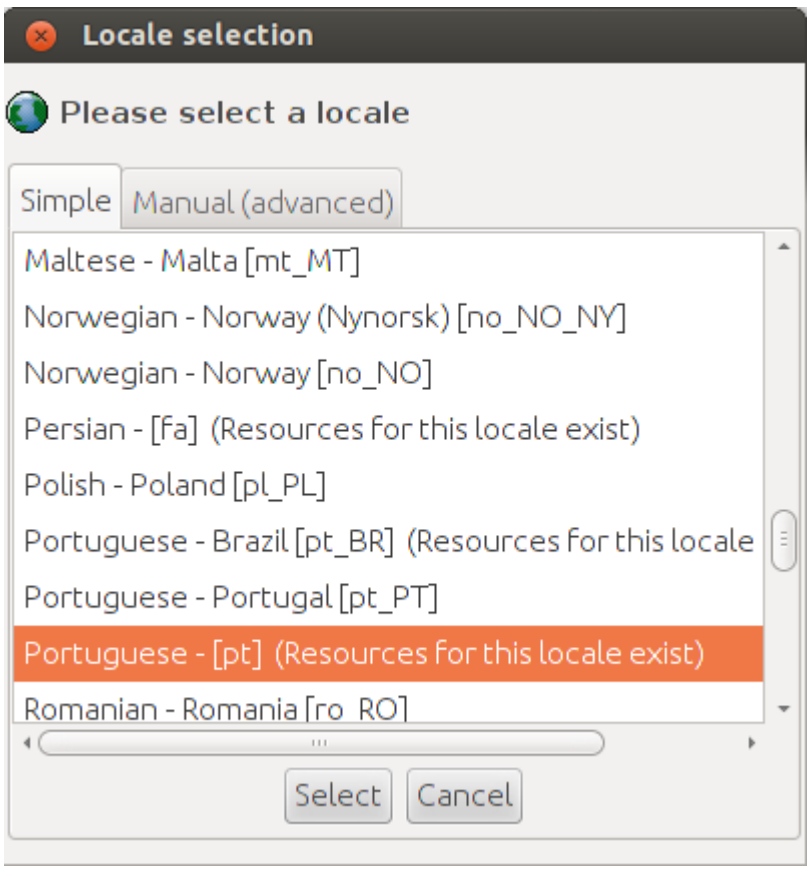
The i18n resource editor is Java desktop application which can be downloaded from http://www.dhis2.org/downloads . It requires that you have checked out the DHIS2 source code from Github (check out http://www.dhis2.org/development if necessary) and have a Java Runtime Environment installed on your computer. On Windows, simply unpack the ZIP archive and click the executable file. On Linux, extract the archive, navigate inside the extracted directory and invoke the following command:

```
java -jar dhis-i18n-resourceeditor.jar
```

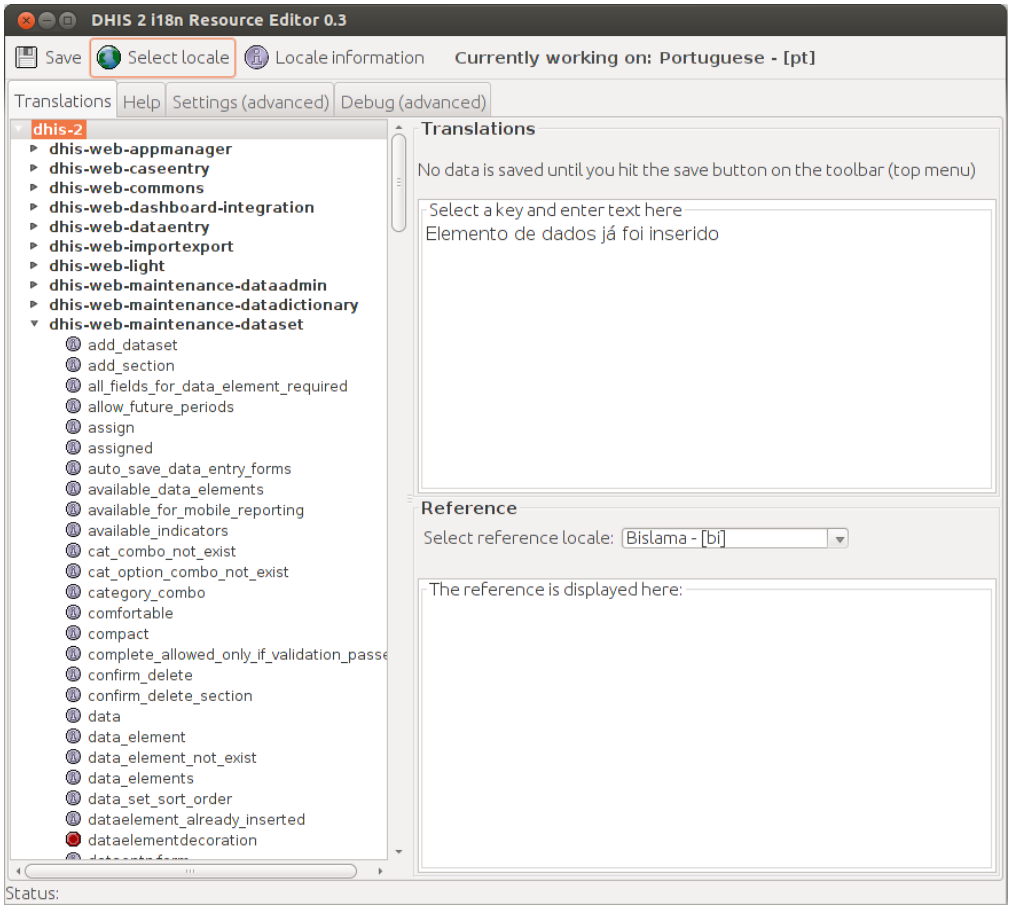
1. Press “Browse” when the application starts and select the path to the “dhis-2” directory inside your local copy (checkout) of the DHIS 2 source code repository, followed by OK.



2. Next, select the destination locale which you will translate strings into. Remember that if you want to create or add to a general language translation, select e.g. “Portuguese - [pt]. If you want to create a country-specific translation, select e.g.”Portuguese - Brazil [pt_BR]. Locales which already have keys translated will show the text “Resources for this locale exist”.



3. Select one of the web modules from the left hand side to translate, e.g. dhis-web-maintenance-dataset.



   Once you have selected a module, click on a particular key from the left-hand side. A reference value for the key will be displayed in the lower right-hand pane, and the translation value will be displayed in the upper right-hand pane. Keys with missing values will be indicated with a red icon. If the value does not exist, simply add the translation there.

4. Once you have finished translating, make sure to press the “Save” button.

5. Once you have finished all of the translations, please submit a pull request on the appropriate GitHub repository. A member of the development team will review the translations for eventual incorporation into the main source code.

20.3 Using the DHIS 2 translation portal

A web-based portal solution has been setup in order to facilitate the translation of DHIS 2 into multiple languages. Simply direct your browser to https://translate.dhis2.org/ and register for an account by providing a username, email address and password. The server will send you a confirmation email which you can use to activate your account. Once you have activated your account, simply press the “Log in” link from the main portal page, and provide your username and password.

The first time you login, you should select your settings, by clicking “My account->Settings”. Here you can select your interface language, the projects which you wish to work on, and the languages which you will translate into. Be sure to press “Save” when you have finished making your changes.

To start translating, be sure you have logged in, and then press the “Home” link in the upper-right hand corner. Select a project (e.g. DHIS 2) and then click on the language which you wish to

translate. The number of words which need to be translated will be displayed under the "Summary" field. Click on the module you wish to translate and then keep drilling down through the folders to find a module which needs translation, (e.g. dhis-web->dhis-web-caseentry). Now click on the "Summary" text which will say something like "194 words need attention". You will be directed to the string which requires translation.

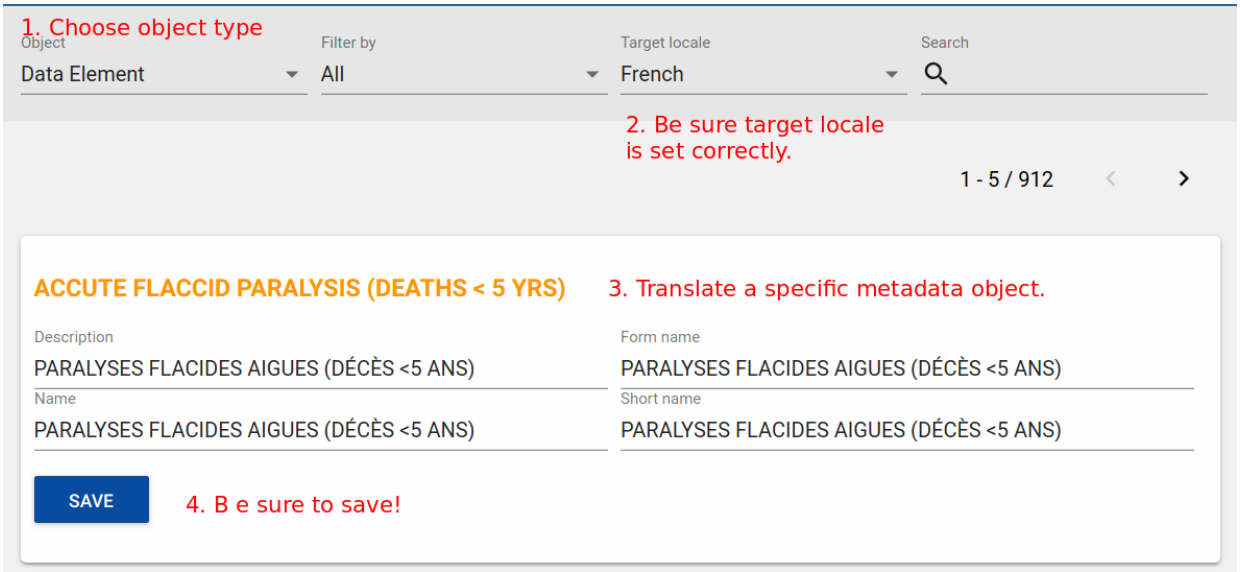
Simply translate the term, and then press "Submit".

Once you have finished translating a module/app, please submit a new issue on https://jira.dhis2.org. Your translations will be incorporated on a regular basis into the main source code of DHIS 2.

20.4 DHIS 2 translation app

In addition to translation of the user interface, DHIS 2 also supports the localization of the metadata content in the database. It is possible to translate individual objects through the maintenance app, but in order to better support a standard translation workflow, a specialized app has been developed for this purpose. The DHIS 2 translation app can be used to translate all metadata (data elements, categories, organization units, etc) into any locale which is present in the database.

To get started, simply choose the "Translations" app from the top level menu.



1. To get started, choose the type of object you wish to translate from the drop-down menu, such as “Data elements”.
2. Next, be sure you have set the “Target locale” to the correct language.
3. Choose the specific object you wish to translate, and translate each of the properties (Name, Short name, Description, etc). These properties will vary from object to object.
4. Press “Save” when you are done translating the specific object to save your changes.

Note that you can search for a specific term using the search feature in the upper right hand corner of the app.

21 DHIS2 Tools Guide

21.1 Overview

The dhis2-tools package is a collection of tools and utilities for installing and managing DHIS2 applications on an ubuntu server. The tools provide the ability to go from a “blank” server with only ssh running, to a fully functioning dhis2 installation in a matter of minutes. Used together they can also be combined into automated scripts to facilitate rapid reconstruction of a given configuration.

The tools have been collected and developed over a number of years. This documentation differs in some respects from the installation guidelines in the dhis2 user manual in that it describes the implementation of a specific approach rather than the more general tutorial nature of the user manual. It is recommended that implementers do also study the material in the user manual as it provides additional information, eg. how to tune the postgresql server. The rationale of the tools described in this manual includes:

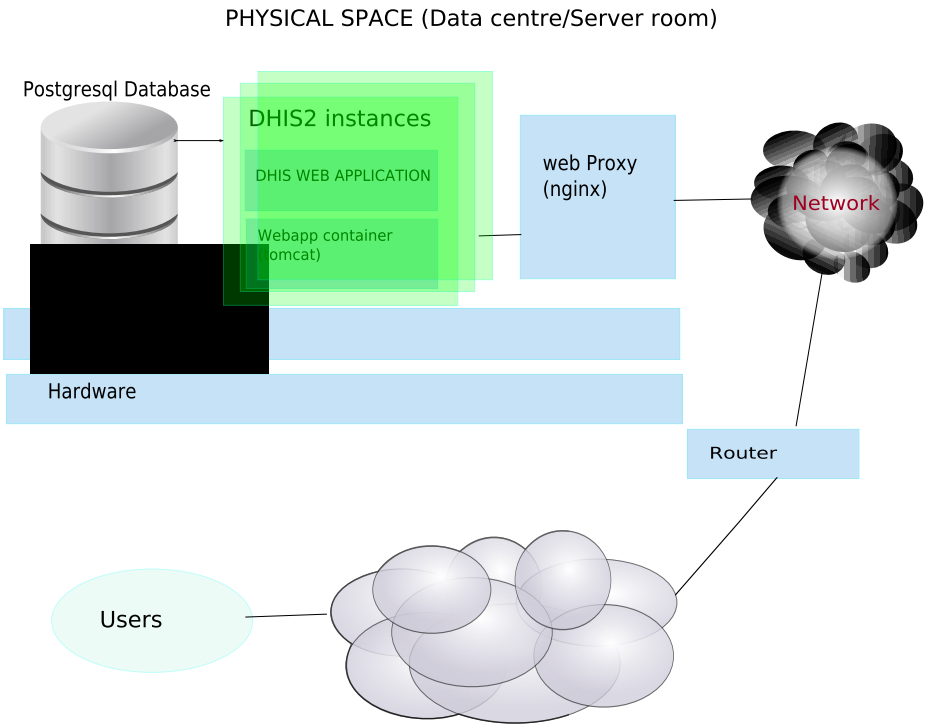
1. to ease the process of installation so that it can be easily explained, documented and executed;
2. to assist system administrators (particularly, but not exclusively, lesser experienced ones) to implement reasonable security measures by default and thus minimize vulnerabilities brought about through human error and negligence;
3. to provide a set of scripts to assist the administrator with tasks related to managing their dhis2 system, beyond the one off process of installation.

The package remains a work in progress and there are a number of areas where it could and should (and hopefully will) be improved. For example,

1. currently the tuning of the postgresql database is not covered. There are ways in which this could be at least semi automated;
2. nginx configuration is assisted by means of providing a sample configuration file. This configuration could be made more dynamic;
3. the format of what is currently packaged is an Ubuntu linux deb package. There is also considerable interest in the Redhat/CentOS flavour of linux for running dhis2. It should be possible to offer a yum format package to facilitate use on these systems.

21.2 Architecture

The figure below shows the main components involved in a DHIS2 system.



Single machine all-in-one installation

The dhis2-tools are primarily concerned with the creation and managing of the tomcat instances which deliver the web application. As you can see in the diagram there may be one or more of these. In addition the system requires a postgresql database server and an nginx web proxy server. There are many possible configurations where these can be running on a different server than the dhis2 instances. These tools for the most part assume that they are all installed together on the one machine. Some customisation is required to separate them.

When an instance is created with the dhis2-create-instance command, a new user is created with the name of the instance (lets say its called hmis). The home directory of that user is located at /var/lib/dhis2/hmis. A database role is created also called hmis together with a database with the name of hmis. The DHIS2_HOME environment variable for the instance is set to the same home directory of the user.

The essential components of a standalone tomcat instance are also created within the same directory (modelled after the ubuntu tomcat7-user package). The web.xml file of that tomcat instance has been customized to allow an upstream web proxy server (such as nginx) to cache the static content of the dhis2 application.

The user will also have a crontab configuration automatically setup to manage daily backups, start on computer restart and log file rotation.

Note that postgresql optimization, as described in the dhis2 user documentation, is not managed by this package and needs to be done as a post-installation step.

21.3 Installation

This manual assumes that you have installed a minimal distribution of ubuntu server 12.04 LTS, 14.04 LTS or 16.04 LTS. By minimal we mean that only the base operating system is installed together with an openssh server. During the installation you should avoid to install ANY other packages The dhis2-tools package will ensure that the required packages are installed as dependencies.

It is recommended as a general guideline that before proceeding any further you should strengthen the security of the system at this point by improving the security of your ssh service and installing a host based firewall like ufw.

Once your base system is properly installed and secured you can proceed to install the dhis2-tools package from the PPA repository at https://launchpad.net/~simjes91/+archive/ubuntu/dhis2-tools. The easiest way to do so is to run the install.sh script available (with the source code of the package) at https://github.com/dhis2/dhis2tools.

The simplified set of steps to get a dhis2 instance up and running from here are:

1. turn your user (eg bobj) into a dhis2-admin user by running:

   sudo dhis2-create-admin bobj

2. create an instance named eg dhis with:

   dhis2-instance-create dhis

3. deploy the latest stable war file with:

   dhis2-deploy-war dhis

4. setup a basic nginx template with:

   dhis2-nginx

   Note that nginx configuration is not done automatically. Though running the command dhis2-nginx will create a simple site configuration file under /etc/nginx/sites-enabled/dhis2. You may need to edit this file to ensure that instance names and port numbers are correct.

5. start your dhis instance with:

   dhis2-startup dhis

A full description of these commands and others used for managing your instance is included in the command reference section below.

21.4 DHIS2 tools reference

The reference documentation for the commands contained in the package is listed in the pages below. This documentation should also be included as man pages when the package is installed. So for example you should be able to type

```
man dhis2-instance-create
```

to read the documentation for that command on the system. Typing

```
apropos dhis2
```

will show you all the dhis2 related man pages.

Name: dhis2-instance-create
Purpose: Creates a new dhis2 instance

` /usr/bin/dhis2-instance-create

[OPTIONS]

name `

Use this tool to create a new dhis2 instance in a tomcat container. The name that is specified will be used to create a new user and a new database with the name of that user. The user will be assigned to the **dhis2** group. The user will have a home directory created in /var/lib/dhis2/<username>. This directory acts as both the **DHIS2_HOME** directory and also the **CATALINA_BASE** directory for the tomcat servlet container.

By default the instance is allocated 2G of heap space RAM. This can be adjusted by editing the parameters in **/var/lib/dhis2/<name>/bin/setenv.sh**.

The servlet container is configured to run with an http connector pool of a maximum of 100 threads. This parameter can be adjusted by editing **/var/lib/dhis2/<name>/conf/server.conf**.

The servlet container configuration has been specially tweaked for running DHIS2. For example tomcat filters are used to ensure that all static content from the web application are cacheable by web proxy servers such as nginx or apache. The lib directory of the webapp has been explicitly placed in the application classpath so that additional jars such as java compiled apache camel routes can be made available to the DHIS2 application.

Note that a dhis2 war file is not deployed by default. See the manual page for **dhis2-deploy-war** for instructions to deploy a dhis2 war file over the internet from the latest stable global build, latest trunk build or from a user specified war file on the filesystem.

You need to be a member of the **dhis2-admin** group to use these and other tools for managing the instance. See the manual page for **dhis2-create-admin**.

- -p
  http port
- -n
  DO NOT create the database when creating the instance. Note if you use this option you will have to manually edit the properties file at /var/lib/dhis2/<instance>/dhis.conf.

```
dhis2-instance-create -p 8080 hmis
```

Creates a new instance called hmis listening on http port 8080.

dhis2-create-admin (1), dhis2-deploy-war (1), dhis2-startup (1), dhis2-shutdown (1), dhis2-deploy-war (1) and dhis2-log (1).

Name: dhis2-startup
Purpose: Starts a dhis2 instance

```
/usr/bin/dhis2-startup instance name
```

Start a dhis2 instance

dhis2-startup myInstance

dhis2-shutdown (1), dhis2-deploy-war (1) and dhis2-instance-create (1).

Name: dhis2-shutdown
Purpose: Stops a dhis2 instance

```
/usr/bin/dhis2-shutdown instance name
```

Stop a dhis2 instance

dhis2-shutdown myInstance

dhis2-startup (1)

Name: dhis2-clone
Purpose: Clones the database of one instance to another instance

```
/usr/bin/dhis2-clone master copy
```

This command creates a copy of the database and war file of one instance into another instance. The main use case for this is where you want to setup an instance for training purposes. Trainees can be “let loose” on the training instance without fear of disturbing the data or configuration of the production instance. They will however be working with the same usernames, forms and reports which exist in the master. The command should be executed with care as it will completely replace the existing database of the target instance

Scheduled datamart and analytics generation jobs are disabled in the target instance.

The command could conceivably be scheduled to run in the early morning to ensure that the database is restored to a pristine state for the start of each day’s training. Or it can be run on demand.

dhis2-clone hmis training

Creates a new instance called training from an existing instance called hmis.

Name: dhis2-deploy-war
Purpose: Deploys a war file

` /usr/bin/dhis2-deploy-war

[OPTIONS]

instance name `

- -t
  Deploy war from latest trunk build. NOT RECOMMENDED for production systems
- -l
  Deploy war located at a custom url
- -f
  Deploy war from a file on the filesystem

Deploys a dhis2 war file to the instance. The default behaviour when no options are given is to download and deploy the latest stable release from http://stable.dhis2.org.

`dhis2-deploy-war myInstance` deploys the latest stable release from dhis2.org into myInstance.

`dhis2-deploy-war -f wars/dhis.war myInstance` deploys the war file at wars/dhis.war into myInstance.

`dhis2-deploy-war -t myInstance` deploys the latest trunk build from the dhis2 team integration server into myInstance. Don’t use this in production.

`dhis2-deploy-war -l http://mywars.org/dhis.war myInstance` deploys the war file from a user provided url into myInstance.

Name: dhis2-logview
Purpose: Shows log file

```
/usr/bin/dhis2-logview instance name
```

Use this tool to view log of dhis2 instance using less. Type ":q" to exit. See the man page for less for tips in navigating and searching the file.

dhis2-logview myInstance

dhis2-logtail (1).

Name: dhis2-logtail
Purpose: Shows the bottom log file in real time. Type Ctrl-C to exit.

```
/usr/bin/dhis2-logtail instance name
```

Use this tool to show the log of dhis2 instance in real time.

dhis2-logtail myInstance

dhis2-logview (1).

Name: dhis2-create-admin
Purpose: Create a user for administering dhis2 instances

```
/usr/bin/dhis2-create-admin username
```

Creates a new dhis2 admin user. If the specified user does not exist, she will be created on the system. Otherwise an existing user is modified. The dhis2 admin user will have postgres superuser privileges and wil be a member of the dhis2admin group.

Create it like this

Name: dhis2-nginx
Purpose: Configure nginx with a specified or sample file

` /usr/bin/dhis2-nginx

[FILENAME]

`

Use this tool to configure nginx. If no file is specified, a sample file will be used.

The sample file is located at **/usr/share/dhis2-tools/samples/nginx/**

```
dhis2-nginx
```

Configures nginx to use the sample configuration.

Name: dhis2-integrity
Purpose: Check database integrity of a DHIS2 instance

` /usr/bin/dhis2-integrity

[instance name]

`

Use this tool to check the integrity of a dhis2 database. The tool runs multiple sql queries to the specified database.

```
dhis2-integrity dhis
```

Runs various integrity tests on the database named dhis

Name: dhis2-restoredb
Purpose: Restore a database dump to a dhis2 instance.

` /usr/bin/dhis2-restoredb

[instance name]

[db dumps]

`

Use this tool to restore a database.

Shuts down the specified dhis2 instance and takes a snapshot backup of the current database. It then drops the current database and creates a new blank one. The new database is then populated with the specified db dump.

```
dhis2-restoredb myInstance db.sql
```

Restores the db.sql dump to the myInstance database

Name: dhis2-backup
Purpose: Create a backup of a dhis2 database

```
/usr/bin/dhis2-backup
```

Use this tool to create a backup of a database.

```
dhis2-backup
```

Creates a backup in the ~/backup folder

Name: dhis2-delete-instance
Purpose: Deletes the specified DHIS2 instance and its database.

` /usr/bin/dhis2-delete-instance

[OPTIONS]

name `

Use this tool to delete a DHIS2 instance. This will delete the user along with its home directory. It also deletes the database user and the database.

- -d
  database name

```
dhis2-delete-instance -d dhisdb dhis
```

Deletes a dhis2 instance named 'dhis' and its database named 'dhisdb'

21.5 Troubleshooting guide

The following table shows some common problems which occur and likely remedies:

Troubleshooting guide

Problem	Solution
When you attempt to access the site with your browser it does not connect.	Either there is a network problem or nginx is not running. Check first to see if you can ping the host. If not you have a network problem. If you can ping the site, the most likely problem is that nginx is not installed or is not running. Verify that nginx is up and running and listening on ports 443 and 80 by typing: `sudo netstat -ntlp` You should see the nginx process listening on those 2 ports
You can access the site but you see a 502 gateway error in your browser.	This means that nginx is unable to connect to your backend dhis2 instance. Either the instance is not running or your nginx location configuration has an error. Running the same netstat command above should show your instance listening on 127.0.0.1 with a port number typically 8080 or whatever you have configured it as. If its not running, try to start it with `dhis2-startup [instance name]` If it is still not running, check the log file with `dhis2-logview [instance name]` to see if there is any information indicating why it has failed to start. If it is running and you can see it with netstat then you need to check your nginx configuration file to ensure that the locatio is correctly mapped.

Problem	Solution
You can access the site but you see a blank page in your browser.	This usually means that the dhis2 instance is running, but you have forgotten to deploy a war file to it. You need to run dhis2-deploy-war on that instance. See the reference section above for details of options.

22 DHIS 2 Documentation Guide

22.1 DHIS 2 Documentation System Overview

DHIS 2 is a web-based information management system under very active development with typically three releases per year. Each release typically includes a number of new features and additional functionality. Given the fast pace of development, the system's wide user base and distributed, global nature of development, a comprehensive documentation system is required.

In this chapter, we will describe the documentation system of DHIS 2 and how you can contribute.

22.2 Introduction

The DHIS 2 documentation is written in Commonmark markdown format. One of the main advantages of markdown is that there is complete separation between the content and presentation. Commonmark is a strongly defined, highly compatible specification of markdown. Since markdown can be transformed into a wide variety of formats (HTML, PDF, etc) and is a text-based format, it serves as an ideal format for documentation of the system.

There exist a wide range of text editors which can be used for the creation of markdown files. For Linux and Windows, ghostwriter is a nice option; it is free and supports side-by-side preview and custom style sheets. One of the key concepts to keep in mind when authoring documentation in markdown, or other presentation neutral formats, is that the **content** of the document should be considered first. The **presentation** of the document will take place in a separate transformation step, whereby the source text will be rendered into different formats, such as HTML and PDF. It is therefore important that the document is well organised and structured, with appropriate tags and structural elements being considered.

It is good practice to break your document in to various sections using the section headings. In this way, very complex chapters can be split into smaller, more manageable pieces. This concept is essentially the same as Microsoft Word or other word processing programs. The rendering process will automatically take care of numbering the sections for you when the document is produced.

22.3 Getting started with GitHub

The DHIS 2 documentation system is managed at GitHub in its own source code repository. GitHub is a collaborative platform that enables multiple people to work on software projects collaboratively. In order for this to be possible, a version control system is necessary in order to manage all the changes that multiple users may make. GitHub uses the *git* source control system. While it is beyond the scope of this document to describe the functionality of *git*, users who wish to create documentation will need to gain at least a basic understanding of how the system works. A basic guide is provided in the next section. The reader is referred to the git manual for futher more detailed information.

In order to start adding or editing the documentation, you should first perform a checkout of the source code. If you do not already have a GitHub account, you will need to get one. This can be done here. Once you register with GitHub, you will need to request access to the *dhis2-documenters* group if you wish to modify the source code of the documentation directly.

Login to GitHub, and then file an issue here. Your request will need to be approved by the group administrators. Once you have been granted access to the group, you can commit changes to the documentation branch and send and receive notifications if you wish. Alternatively, you can clone the documentation into your own repository, commit your changes to your own fork, and

request that your changes be merged with the source of the documenation with a pull request here.

22.4 Getting the document source

In order to edit the documentation, you will need to download the source of the documentation to your computer. GitHub uses a version control system known as git . There are different methods for getting Git working on your system, depending on which operating system you are using. A good step-by-step guide for Microsoft operating systems can be viewed here. Alternatively, if you are comfortable using the command line, you can download git from this page If you are using Linux, you will need to install git on your system through your package manager, or from source code. A very thorough reference for how git is used is available in a number of different formats here.

Once you have installed git on your system, you will need to download the document source. Just follow this procedure:

1. Make sure you have git installed.
2. On Windows systems, visit https://github.com/dhis2/dhis2-markdown-docs and press “Clone in Desktop”. If you are using the command line, just type `git clone git@github.com:dhis2/dhis2docs.git`
3. The download process should start and all the documentation source files will be downloaded to the folder that you specified.
4. The DHIS 2 documents depend on other branches for their documentation. Be sure to keep these these up to date as well. When you build the documentation, the necessary sub-modules will be downloaded automatically as part of the build process ,if you have not already done so.
5. Once you have the source, be sure to create your own branch for editing. Simply execute`git checkout -b mybranch` where *mybranch* is the name of the branch you wish to create.

22.5 Editing the documentation

Once you have downloaded the source, you should have a series of folders inside of the repository directory. The documents are structured as follows:

```
<root>
└── src
    └── commonmark
        └── en
            ├── dhis2_android_user_man_INDEX.md
            ├── dhis2_developer_manual_INDEX.md
            ├── dhis2_end_user_manual_INDEX.md
            ├── dhis2_implementation_guide_INDEX.md
            ├── dhis2_user_manual_en_INDEX.md
            ├── user_stories_book_INDEX.md
            ├── resources
            │   ├── css
            │   │   ├── dhis2.css
            │   │   └── dhis2_pdf.css
            │   └── images
            │       └── dhis2-logo-rgb-negative.png
            └── content
                ├── android
                │   └── resources
```

```
│       └── images
├── common
│   └── bookinfo.yaml
├── developer
│   ├── resources
│   │   └── images
├── implementation
│   ├── resources
│   │   └── images
│   ├── `resources
│   │   └── images
├── stories
│   ├── resources
│   │   └── images
└── user
    └── resources
        └── images
```

The `*_INDEX.md` files are the starting points for the master documents. They contain only `!INCLUDE` directives.

e.g. dhis2_android_user_man_INDEX.md:

```
!INCLUDE "content/common/about-this-guide.md"
!INCLUDE "content/android/configure-dhis2-programs-to-work-on-android-apps.md"
!INCLUDE "content/android/android-event-capture-app.md"
!INCLUDE "content/android/android-aggregate-data-capture-app.md"
!INCLUDE "content/android/android-tracker-capture-app.md"
```

The `!INCLUDE` directives point to the “chapters” that are used to make up the manual.

> **NOTE**
>
> the `!INCLUDE` directives are not part of pure commonmark format, but are used in pre-processing to build the master documents. The particular format here is the one supported by markdown-pp out of the box, but we could change it to another “include” format if desired.

It is perfectly valid to use `!INCLUDE` directives in the sub-documents too, but currently the documents are split up at chapter level only.

- For the sake of convention, place all chapters in a folder in the following format:

  `./src/commonmark/en/content/XXXX/<chapter>.md`

  where XXXX represents one of the thematic folders which are used to organize the documentation, and `<chapter>` is the filename referenced from the `*_INDEX.md` file.

22.5.1 Using images

Image resources should be included inside a folder structure beginning with `resources/images/` relative to the current document. e.g. for the chapter `content/android/android-event-capture-app.md`, the images are somewhere under `content/android/resources/images/<rest-of-path>`.

This is important because the `resources/images` string is used to identify images in the files. Images will be collected under `resources/images/content/android/<rest-of-path>` relative to the master document, when the the files are pre-processed for generation. *The paths are partially reversed to ensure they remain unique when collecting images from multiple thematic folders.*

22.5.2 Section references

In order to provide fixed references within the document, we can set a fixed text string to be applied to any section. For our markdown docs this is done by adding a comment after the section heading in the form:

```
<!-- DHIS2-SECTION-ID:name_of_section -->
```

where `name_of_section` is replace with the id you wish to use.

For example:

```

## Validation

<!--DHIS2-SECTION-ID:webapi_validation-->

To generate a data validation summary you can interact ...
```

Will set the section id of the level 2 heading **Validation** to “webapi_validation”, which may then be referenced as “#webapi_validation” within the html file.

After the full html file is generated, it is post-processed and the first `DHIS2-SECTION-ID` after the start of the section is used as the section id.

Please follow the convention of lowercase letters and underscores, in order to create id’s that are also valid as filenames when the html files are split.

22.6 DHIS 2 Bibliography

Bilbliographic references are currently not supported in the markdown version of DHIS 2 documentation.

22.7 Handling multilingual documentation

The DHIS 2 documentation has been translated into a number of different languages including French, Spanish and Portuguese. If you would like to create a translation of the documentation or contribute to one of the existing translations, please contact the DHIS 2 documentation team at the email provided at the end of this chapter.

22.8 Committing your changes back to GitHub

Once you have finished editing your document, you will need to commit your changes back to GitHub. Open up a command prompt on Windows or a shell on Linux, and navigate to the folder where you have placed your documentation. If you have added any new files or folders to your local repository, you will need to add them to the source tree with the `git add` command, followed by the folder or file name(s) that you have added. Be sure to include a descriptive comment with your commit.

```
git commit -m "Improved documentation on organisation unit imports with
CSV."
```

Finally, you should push the changes back to the repository with `git push origin mybranch`, where “mybranch” is the name of the branch which you created when you checked out the document source or which you happent o be working on. In order to do this, you will need the necessary permissions to commit to the repository.When you have committed your changes, you can issue a pull request to have them merged with the master branch. You changes will be reviewed by the core documentation team and tested to ensure they do not break the build, as well as reviwed for quality. As mentioned previously, you can also push your changes to your own GitHub repo, if you do not have access to the main repo, and submit a pull request for your changes to be merged.

If you have any questions, or cannot find that you can get started, just raise a question on our development community of practice.

23 Using JIRA for DHIS2 issues

23.1 Sign up to JIRA - it's open to everyone!

1. Go to: https://jira.dhis2.org.
2. Create an account with your name and email address.

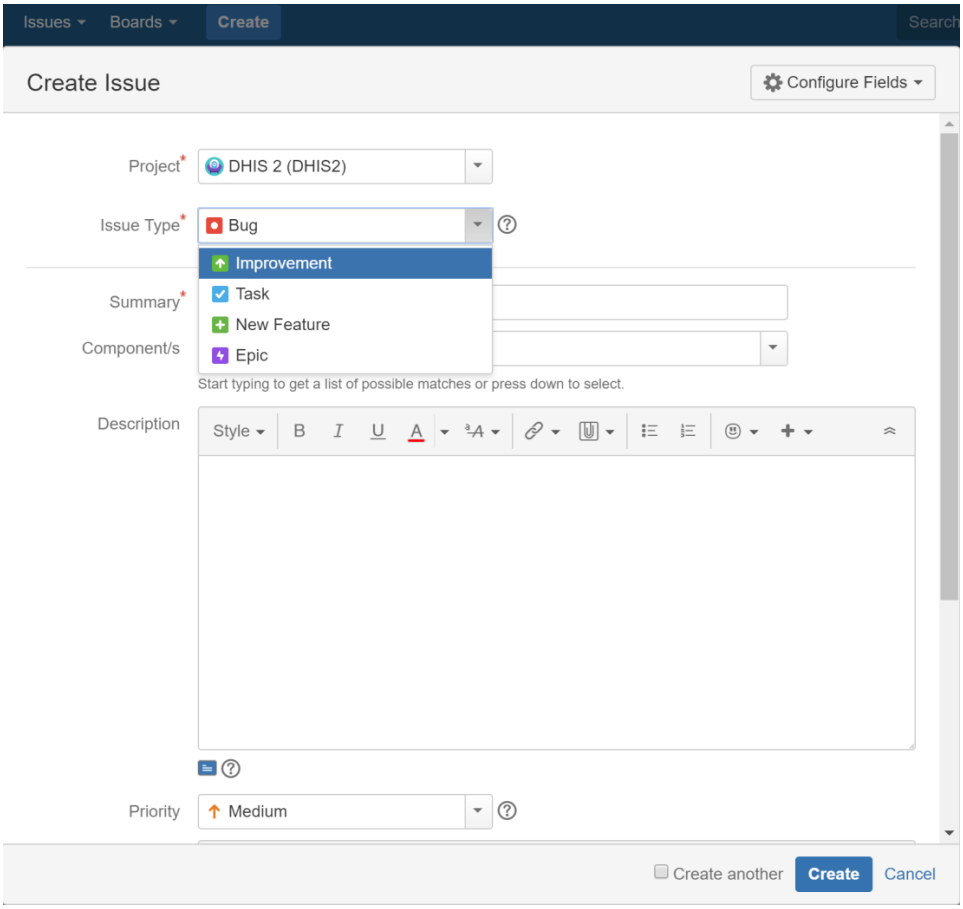
23.2 Report an issue

> **Tip**
>
> Uncertain whether something is a missing feature, a bug or deprecated? We'd really appreciate that you ask on the developer list before reporting a bug directly. Thanks!

1. Click **Create** in the top menu.
2. Select a **Project** from the list.
3. Select an **Issue Type**:
    - **Improvement** - if you'd like to tell us about something that could be better such as usability or design suggestions.
    - **New feature** - if you want to suggest a feature.
    - **Task** - if you've been asked to work on a DHIS2 task.
    - **Bug** - if you've found something that needs fixing.
    - **Epic** - if you'd like to submit an idea for a new DHIS2 area such as an app. Epic is used for issues more complex than new features.
4. Click **Create**.

> **Tip**
>
> To create several issues in one go, select **Create another**.

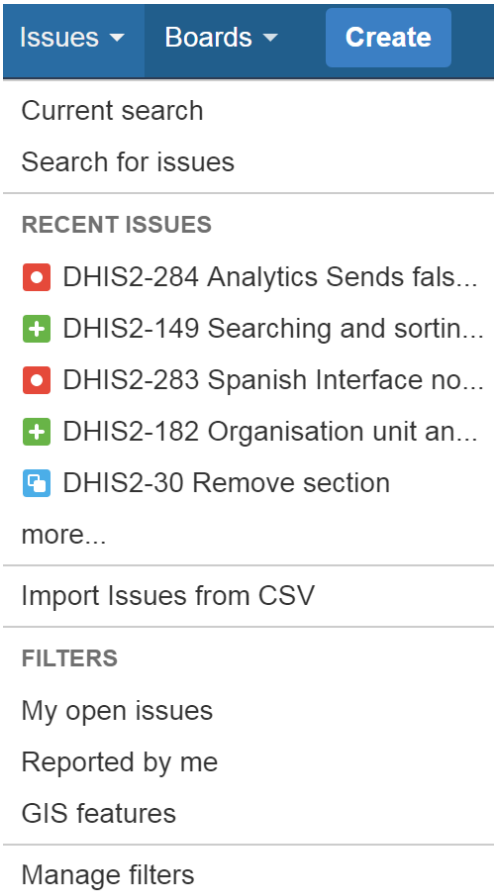
5. Fill out the issue form. Please give us plenty of context! Include server logs, JavaScript console logs, the DHIS2 version and the web browser you're using.



23.3 Search for issues

Click **Issues > Search for issues**.

If you click **Advanced**, you can order your search criteria using the predefined fields. You can also enter search terms in the search field. See also: Search syntax for text fields.

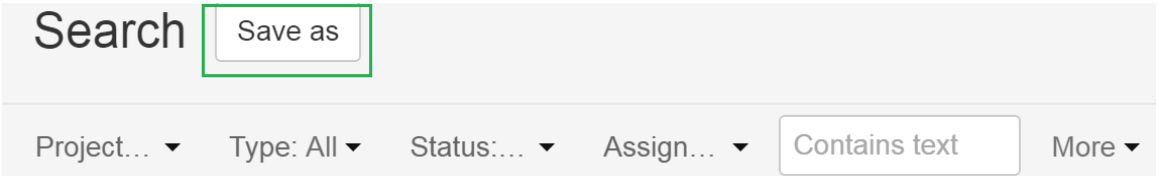


23.4 About filters

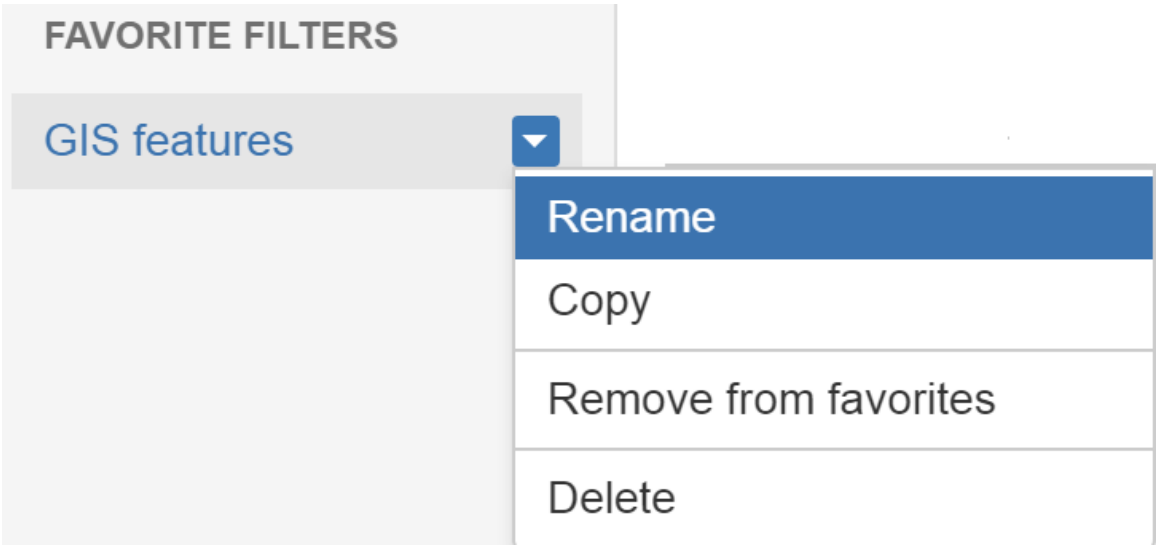
You can save your search results as filters to get back to them faster. A filter is very similar to a favorite. We've created filters for features intended for release numbers 2.26, 2.27 and 2.28 and for all open bugs.

23.5 Create a filter

1. To create a filter, go to **Issues > Search for issues**.
2. Add filters to your search such as:
    - **Project** – the main software project is DHIS 2.
    - **Type** – you can filter by **Standard Issue Types** such as New Features, Improvements, Bugs or Epic or **Sub-Task Issue Types**.
    - **Status** – filter by **To Do** (not started) and **Done**, for example.
    - **Version** - click **More > All Criteria > Fix version** to filter by DHIS2 release number.
    - **Internal**- click **More > Internal feature** to exclude low-level (back end) features from your search.
3. Click **Save As**. This button is at the top of the Search pane.

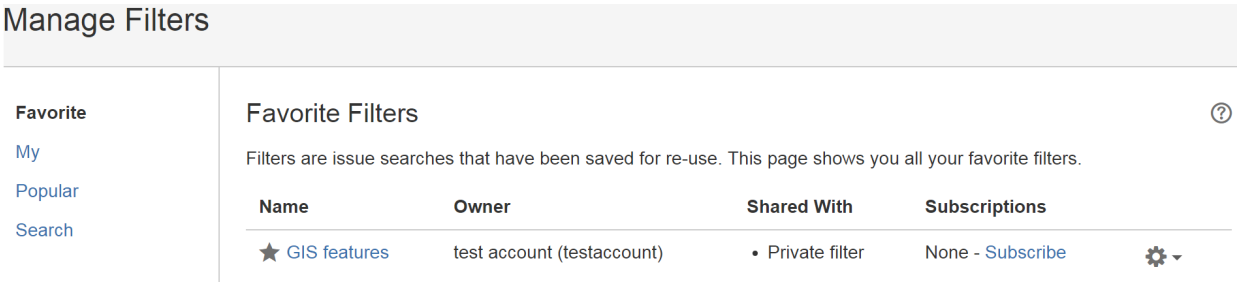


4. Enter a name for your search filter and click **Submit**. Your filter is now available in **Favorite Filters**. Use the arrow to modify your filter. Your filter is also available on the **System Dashboard**.



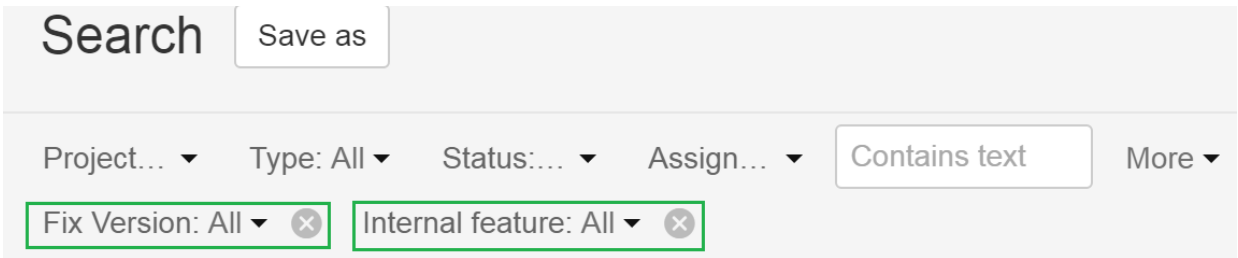
23.6 Add a filter to your profile

To add a filter to your profile, click **Issues > Manage filters** and click the star icon next to each filter.



23.7 Remove search filter terms from your search

Click the cross to remove search filter terms you previously added from your search.



23.8 Communicate with us

To share information, clarify requirements, or discuss details about an issue, do this using issue comments.

1. Select the issue you want to comment.
2. In the Issue Detail view click **Comment** and enter your text.

   To email others about your comment, simply enter **@User's Name** in the comment field. An email will be sent to the users' email addresses that are registered with their JIRA accounts.

3. Click **Add**.