

# DiSSCo related output

This template collects the required metadata to reference the official Deliverables and Milestones of DiSSCo-related projects. More information on the mandatory and conditionally mandatory fields can be found in the supporting document 'Metadata for DiSSCo Knowledge base' that is shared among work package leads, and in Teamwork > Files. A short explanatory text is given for all metadata fields, thus allowing easy entry of the required information. If there are any questions, please contact us at [info@dissco.eu](mailto:info@dissco.eu).

## Title

Milestone Report MS 5.6 "A functional prototype of DiSSCo Modelling Framework"

## Author(s)

David Fichtmueller  
Anton Güntsch

## Identifier of the author(s)

<https://orcid.org/0000-0002-0829-5849> (DF)  
<https://orcid.org/0000-0002-4325-4030> (AG)

## Affiliation

Botanic Garden and Botanical Museum (BGBM)  
Berlin, Freie Universität Berlin

## Contributors

Mareike Petersen <https://orcid.org/0000-0001-8666-1931>  
Marcus Ernst

## Publisher

DiSSCo Prepare

## Identifier of the publisher

## Resource ID

<https://doi.org/10.34960/wn2h-9g16>

## Publication year

2021

## Related identifiers

## Is it the first time you submit this outcome?

Yes

## Creation date

09/08/2021

## Version

1

## Citation

Fichtmüller D. & Güntsch A. (2021) Milestone Report MS 5.6 "A functional prototype of DiSSCo Modelling Framework". <https://doi.org/10.34960/wn2h-9g16>

## Abstract

This DiSSCo Prepare Milestone Report MS 5.6 "A functional prototype of DiSSCo Modelling Framework" describes the technical setup of the DiSSCo Modelling Framework, including the underlying software packages, its core components and their interactions. We highlight relevant modifications that were done to the setup in order to better fulfill the requirements of the Modelling Framework. In the end we will take a look at the issues and challenges that arose during the task so far and have an outlook on the upcoming steps to take the functional prototype to the production installation of the Modelling Framework.

## Content keywords

scientific

**Project reference**  
DiSSCo Prepare (GA-871043)

**WP number**  
WP5

**Project output**  
Milestone report

**Deliverable/milestone number**  
5.6

**Dissemination level**  
Public

**Rights**

**License**  
Attribution 4.0 International (CC BY 4.0)

**Resource type**  
Text

**Format**  
PDF

**Funding Programme**  
H2020-INFRADEV-2019-2

**Contact email**  
d.fichtmueller@bgbm.org



# A functional prototype of DiSSCo Modelling Framework

DiSSCo Prepare WP 5 – Milestone 5.6

David Fichtmueller, Anton Güntsch

**Task Lead**  
BGBM Berlin



---

## Abstract

This DiSSCo Prepare Milestone Report MS 5.6 “A functional prototype of DiSSCo Modelling Framework” describes the technical setup of the DiSSCo Modelling Framework, including the underlying software packages, its core components and their interactions. We highlight relevant modifications that were done to the setup in order to better fulfill the requirements of the Modelling Framework. In the end we will take a look at the issues and challenges that arose during the task so far and have an outlook on the upcoming steps to take the functional prototype to the production installation of the Modelling Framework.

## Keywords

DiSSCo, Modelling Framework, DMF, Wikibase, Wikidata, Docker, Semantic Modelling, Standard Development, Structured Data

# Index

Abstract .....	2
Keywords .....	2
01 Introduction.....	4
DiSSCo Modelling Framework .....	4
Wikidata and Wikibase.....	4
Docker .....	5
02 Methodology .....	6
System Requirements .....	6
Choosing Wikibase .....	7
Components of the Wikibase Setup.....	8
Customizations .....	8
Sharing the Customized Configurations .....	10
Installation.....	10
Conceptual Work.....	12
03 General Issues and Challenges .....	14
Restructuring of Wikibase Docker Container Infrastructure.....	14
Visual Editor.....	14
04 Next Steps.....	15
Starting the Production Instance.....	15
Integration of additional tools.....	16

# 01 Introduction

---

## DiSSCo Modelling Framework

The interoperability of DiSSCo RI software components and services highly depends on the availability of an agreed data model covering all aspects of specimen-related information, for example, specimen and observational data, taxon names, DNA sequences, and publications. The existing landscape of bio- and geodiversity data formats and standards is heterogeneous and modelling approaches are hindered by the current use of different modelling workflows for similar or identical data domains, incomplete version histories, insufficiently documented concepts, and the lack of machine-readable links to related data definitions. The DiSSCo Modelling Framework (DMF) will provide modelling capabilities for required groundworks in order to shape DiSSCo Digital Specimen Object Specifications (DSOS).

Specific use cases range from semantic annotation of collection-related research data to general semantically annotated DiSSCo APIs. Based on an existing technical platform (e.g. Wikidata) and existing standards and data formats developed by the community (e.g. DwC, ABCD, CDM), the modelling framework will provide a mechanism for agreeing on and documenting a core set of “DiSSCo data elements”. The Platform will provide persistent ID for each concept, relations to elements in existing standards, as well as an API for machine-readable access to the data model.

## Wikidata and Wikibase

Wikidata<sup>1</sup> is a free and open knowledge base for structured data. It is a project of Wikimedia, the organization that also runs Wikipedia. Wikidata was started in 2012 and is developed by the German chapter of Wikimedia, the Wikimedia Deutschland e.V. There are currently around 94 million items described in Wikidata and all of the data is licensed under the Creative Commons Zero license, making it freely available for anybody to use for any purpose.

Wikibase is the software that runs Wikidata. It is an extension to Mediawiki, which is the same software that runs Wikipedia. Wikibase allows users to create their own knowledge base for purposes that go beyond the scope of Wikidata.

Wikibase has the same user interface as Wikidata and is therefore intuitively easy to use for people who have already interacted with Wikidata. But also for novice users is the editing process quite straight forward.

There are two main types that make up the general data model of Wikibase (and thus also of Wikidata): Items and Properties. Items are the pages where the concepts of Wikibase instance are described using the properties. Both the individual items and the individual properties are identified by a number that is either preceded by a Q for items or by a P for properties. They can be described using a label, a description and multiple aliases, all of which can exist in multiple languages. Additionally properties can be added to each item or property to either link to other items or to add content depending on the datatype of the property, e.g. text, numbers, dates, coordinates, images. A particular combination of item, property and value is called a statement.

---

<sup>1</sup> <https://www.wikidata.org>

There can be multiple statements for a particular property. A statement itself can also have additional statements, that are either qualifiers (e.g. to limit the time frame when a certain statement was true) or references (to show the sources of a statement).

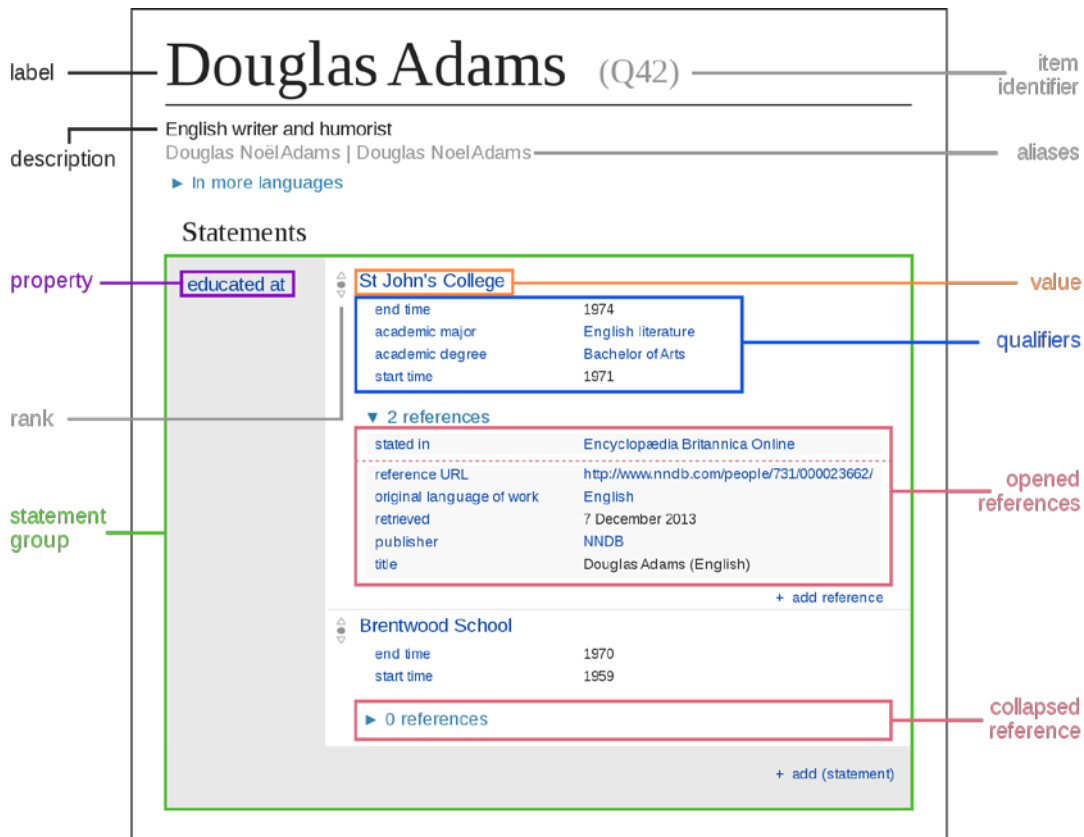


Figure 1: Diagram of a Wikidata Item

Source: Charlie Kritschmar et al., [https://commons.wikimedia.org/wiki/File:Datamodel\\_in\\_Wikidata.svg](https://commons.wikimedia.org/wiki/File:Datamodel_in_Wikidata.svg)

## Docker

Docker<sup>2</sup> is a virtualization platform that allows multiple services (so called containers) to be started and to interact with each other. Dedicated configuration files (Dockerfiles) specify how a container is build and what software it should run. The setup of multiple Docker containers with their individual Dockerfiles and their interactions are configured within a so called docker-compose file.

With Docker it is possible to codify complex server setup, so that new servers with the same configuration can be easily recreated, for instance for load balancing or to swap out individual containers while the rest of the system remains running.

Individual directories within the virtual environment can be specified as volumes that are mapped to directories in the host system. When a container is restarted, the entire container is reset to its initial state, only data stored within a volume are persistent.

<sup>2</sup> <https://www.docker.com>

For the installation of a Wikibase instance a ready-made Docker image is provided by Wikimedia Germany.

## 02 Methodology

---

### System Requirements

Based on the outline of the DiSSCo Modelling Framework in the DiSSCo Prepare work plan the following requirements for the technical basis for the modelling framework were formulated

#### *Collaborative Editing*

The system must be able to allow for multiple users to work on the same standard without additional exchange of files. It should also be possible to work together in real time on the same standard.

#### *User control*

The system must come with a build in user management system including the proper handling of user credentials. Integration into common Single SignOn (SSO) infrastructures is highly desired.

#### *APIs for automated access for mass editing and export*

The software product should have an Application Programming Interface (API) to allow for automated access to external software tools to edit data en masse and to export the data in other formats.

#### *Versioning*

There needs to be a continuous versioning of all the modifications done by any of the users. It needs to transparently log what was modified by whom and when.

#### *Open Source Software*

Because of the commitment of DiSSCo to Open Science, a software solution that is licensed under an open source license is highly desired.

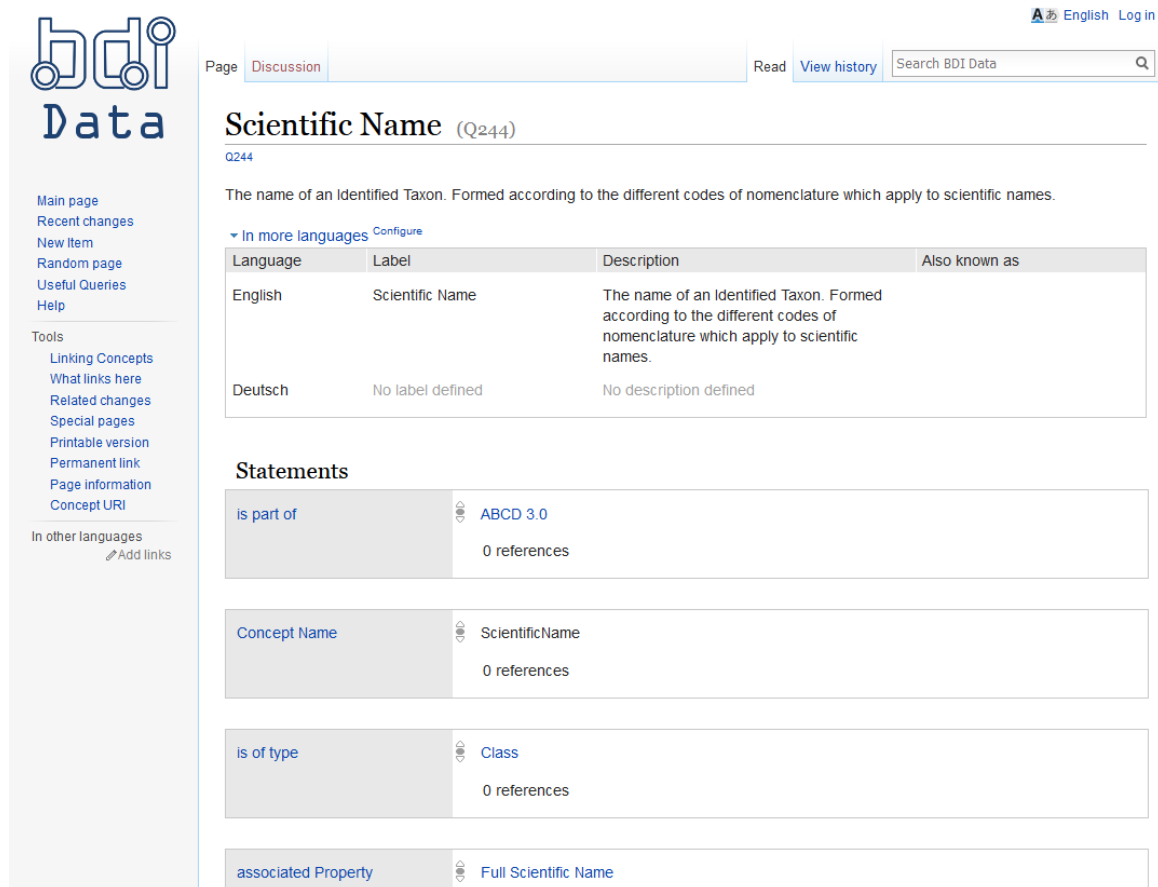
#### *Stable Product with future support*

The software for the DMF needs to be a mature product that is also part of a larger community to ensure future updates, security patches and bug fixes.



## Choosing Wikibase

The teams at the Botanic Garden and Botanical Museum Berlin (BGBM) together with the Museum für Naturkunde Berlin (MfN), both partners in DiSSCo, already had positive experiences using Wikibase for standard development in the context of the ABCD 3.0 project<sup>3</sup>. Because of this we were aware that Wikibase fulfills all of the necessary requirements plus it has the benefit of team that is already experience with installing, running and maintaining a Wikibase instance. In the beginning of the ABCD 3.0 project, a detailed landscape analysis of existing modelling systems was done. In the beginning of DiSSCo Prepare a brief update of this analysis performed to see if any of the reviewed systems had significantly progressed in the meantime. However none of the other systems would come close to the advantages that Wikibase offered. The only real contender worth mentioning was WebProtégé<sup>4</sup>, the web bases version of the popular ontology editor. It has improved significantly in the past years, yet still lacks API access and it is not easy to use for users who might not be ontology experts but domain experts about the modelled topic.



The screenshot shows the Wikibase page for 'Scientific Name' (Q244). The page title is 'Scientific Name (Q244)'. Below the title, there is a description: 'The name of an Identified Taxon. Formed according to the different codes of nomenclature which apply to scientific names.' There is a table with columns 'Language', 'Label', 'Description', and 'Also known as'. The table has two rows: one for 'English' with the label 'Scientific Name' and a description, and one for 'Deutsch' with 'No label defined' and 'No description defined'. Below the table, there is a 'Statements' section with four entries: 'is part of' (ABCD 3.0), 'Concept Name' (ScientificName), 'is of type' (Class), and 'associated Property' (Full Scientific Name). Each entry shows '0 references'.

Figure 2: Screenshot of the Wikibase instance for the ABCD 3.0 Development from the BGBM, <https://wiki.bgbm.org/bdidata/index.php/Q244>

So the decision was made to choose Wikibase as the basis for the DiSSCo Modelling Framework.

<sup>3</sup> Fichtmüller D, Reimeier F, Güntsch A (2019) Using Wikibase as a Platform to Develop a Semantic TDWG Standard. Biodiversity Information Science and Standards 3: e37212. <https://doi.org/10.3897/biss.3.37212>

<sup>4</sup> <https://webprotege.stanford.edu> and <https://github.com/protegeproject/webprotege>

## Components of the Wikibase Setup

### *Wikibase*

The core component of the Wikibase Docker Setup is the Wikibase instance itself. This means it is a Mediawiki instance with the Wikibase Extension installed and configured. It offers the complete functionality of a regular Mediawiki as well as the structured editing of item pages and properties that comes with Wikibase.

The data itself is stored in a Maria database, which is also installed as part of the Docker setup.

### *Query Service*

The QueryService is a service for users to write and execute SPARQL queries on the data of the Wikibase instance. It offers support for writing the queries with syntax highlighting, showing the labels of items and properties used in the query and auto completion. The results are rendered by the Query Service either as a table or as one the other different supported view (e.g. graph, chart, tree), some of which a dependent on the structure or type of the returned data (e.g. time lines, maps, image grid).

The Query Service doesn't access the Wikibase instance directly. It queries the data from a dedicated triple store (Blazegraph). In the Wikibase Docker setup there is an updater script that runs at regular intervals and checks the Wikibase instance for updates. The updates are then imported in Blazegraph. A proxy then sits between Blazegraph and the Query Service that allows only reading access to Blazegraph, as the proxy is publicly accessible.

### *QuickStatements*

QuickStatements is a tool for the easy data import into Wikibase. It uses a very simple syntax of tab separated commands to create or edit properties in items. Once a list of commands is entered, the user can review them and then click on execute. The commands are then executed automatically one after another without the need of intervention by a human user.

This tool has its strength in the mass import and editing of items. The syntax can be easily generated by other tools, such as spread sheets or OpenRefine.

## Customizations

Once the default Docker setup is installed a basic Wikibase instance with all core functionality is available. However for the full desired functionality of the DiSSCo Modelling Framework additional configurations are required, like the installation of additional extensions and modifications to the styling, i.e. the branding. By design most of those changes are lost when the container is restarted, so they need to be made persistent. The best way to do this is to program the customizations into the Dockerfile. This was done for many of the changes to setup. Here is an overview of the most important modifications:

### *Improved customization of components within the docker-compose file.*

Both the Query Service and QuickStatements have the possibility to change the branding name of the components to something custom. This however is in the settings of their respective Dockerfiles. Now it is possible to adjust those names centrally via the docker-compose file.

### *Make QueryService Proxy compatible with OpenRefine*

OpenRefine is a tool that works great in combination with Wikibase, both for producing data to be imported into or for consuming data from Wikibase. The common way of these two products to interact is via a Reconciliation Service, allows the user to connect items from a table in OpenRefine with items from a Wikibase instance. However if one wants to load and edit the output of a Wikibase Query into OpenRefine for cleanup and adjustments (and later export it as commands for QuickStatements), the user needs to run the query manually, save the results in a file and load this file with QuickStatements. With this modification to the Query Service Proxy it is possible to directly run the query URL that is executed by the Query Service from OpenRefine, so that the query is stored with the data and can be easily run again in the future for a later iteration that specific cleanup step.

### *Installation of additional Extensions*

Several additional extensions have been installed to improve the general functionality of the underlying Mediawiki such as SyntaxHighlight to improve code readability or ParserFunctions for improved calculations in template files.

### *Import of useful Templates and Modules*

Over the years the Wikidata community has created powerful templates that improve working with Wikidata. Several of these templates (and the modules that the templates rely on) have been exported from Wikidata, adjusted for use within a generic Wikibase and are now imported during the container installation process, so that they can be used within the custom Wikibase.

### *Move configurations into a persistent folder*

The default file for configurations of a Mediawiki is a file called LocalSettings.php. Most adjustments to any wiki are done in this file. With the Docker setup however, this file would be reset each time the container is restarted. This limits the customizability of the wiki significantly. So a persistent Docker volume was created and LocalSettings.php was moved there. In its place a symbolic link was created that points to the new location, so for the running instance there is no difference.

### *Move sensitive information to dedicated file*

So far the passwords for the various instances are all stored within the docker-compose file. However this becomes a problem when one wants to document the changes made to the Wikibase Docker image and to share it in a public repository. This issue was mitigated by moving the sensitive information into a dedicated environment file from where the sensitive information is automatically extracted and inserted into the docker-compose file when the Docker setup is run.

### *Addition of a web proxy*

In the default configuration all of the individual user-facing services run on different ports, e.g. Wikibase on :8181, Query Service on :8282, QuickStatements on :9191. These ports can be adjusted in the docker-compose file. However if all of these ports should be accessible from one endpoint (i.e. a single public port), a web proxy is required that redirects specific path segments to the dedicated internal ports, e.g. /wiki/ -> :8181

### *Improve QuickStatements to create and edit properties*

On Wikidata the ability to create new properties is limited to administrators who do this after a community consensus about the need for a new property is reached. Since the tool QuickStatements was primarily written for Wikidata and not as a generic tool for Wikibase, it was not possible to create new properties using QuickStatements. Also the editing of properties had some limitations (though this is not due to community limitations). The code of QuickStatements has been adjusted to overcome those limitations. It is now possible to create properties in a Wikibase instance, which is important for mass import of properties.

The patches for these updates were submitted back to the original developer of QuickStatements who has included them into the QuickStatements code base. Therefore it is not necessary anymore to install the custom version of QuickStatements and these modifications are now available for all other Wikibase users.

### *DiSSCo specific settings and branding*

All of the settings that are directly related to DiSSCo have been put into a dedicated settings file which is then included in the LocalSettings.php. The images for the logos and thumbnails were added to the persistent image folder.

## Sharing the Customized Configurations

All of the customizations from the previous chapter (with the exception of the DiSSCo specific settings) are generic enough that other people or organizations running their custom Wikibase instances would benefit from them as well. Therefore the code for the Wikibase Docker setup is kept in two different git repositories for different audiences. There is one with all of the generic updates available at:

<https://github.com/DavidFichtmueller/wikibase-docker/tree/wikibase-docker-improved>

and the one for the DiSSCo specific configurations is available at:

[https://github.com/DiSSCo/modelling\\_framework\\_docker\\_image](https://github.com/DiSSCo/modelling_framework_docker_image)

The latter one is based on the former one, so it is always a few commits ahead.

The changes to the QuickStatements code are not included anymore as the updated QuickStatements will be installed upon the initial setup of the Docker images.

We reached out to the Wikidata/Wikibase development team and asked them if they would consider any of these changes to be included in original repository. Since many of these changes could be an issue for existing installations that need to update their Docker images, we wanted to ask first before just dumping each change as a pull request into their repository. However we have not heard back from them yet. However this might also be due to the *Restructuring of the Wikibase Docker Container Infrastructure* (see next chapter).

## Installation

### *Prototype Instance*

Already at an early stage of the project a Wikibase instance was installed on a server at the BGBM. This instance was further refined and adjusted according to the needs of the DMF. With

the exception of the additional web proxy, it contains all of the adjustments outlined in the previous chapter. The prototype instance is available at: <http://dissco-mf.bgbm.org:8181/wiki/>

The necessary properties required to describe terms of a standard are already defined<sup>5</sup> and users can start to edit pages without the need to create an account or to log in. The wiki is a test bed and playground for future production DMF. Users are free to try, learn and experiment with Wikibase, without affecting any official standards or documentation. The prototype instance will be kept even after the start of the official DMF, as a test bed and sandbox for new users, who want to get used to the Wikibase environment.

The screenshot shows a Wikibase page for 'Demo Term (Q17)'. The page layout includes a sidebar on the left with navigation options like 'Main page', 'Recent changes', and 'Add Item'. The main content area has a 'Discussion' tab selected. Below the title, there is a table of labels in different languages:

Language	Label	Description	Also known as
English	Demo Term	term in the demo vocabulary	
German	No label defined	No description defined	

Below the table, there are sections for 'Statements' and 'additional Description'. The 'Statements' section shows 'is part of' with two entries: 'Demo Vocabulary' and 'Terminology', both with 0 references. The 'additional Description' section shows 'some text (English)' with 0 references.

Figure 3: Screenshot of a Dummy Page on the Prototype Instance of the DiSSCo Modelling Framework: <http://dissco-mf.bgbm.org:8181/wiki/Q17>

### Production Instance

The production release of the DiSSCo Modelling Framework will most likely move to a dedicated subdirectory path of <https://tools.dissco.tech>. The specific domain and the various subdirectory paths need to be set prior to the installation of the Docker instance, as these URLs are then written in the different Docker containers. While changing those paths is possible, it requires a lot of work, especially when a lot of data is already stored as item pages within the Wikibase.

<sup>5</sup> <http://dissco-mf.bgbm.org:8181/wiki/Special:ListProperties>

## Conceptual Work

This section describes how the work with the DMF for the creation and release of a specific standard will be conducted like. It is possible to work on multiple independent standards in parallel in the DMF. However this could cause some issues if the different standards use overlapping vocabulary and similar terms. This is due to the search and autocomplete function when linking to different items, as it shows the best suitable match based on the label and the alias and cannot distinguish between terms belonging to one or the other standard and might result in terms from different and supposedly independent standards that are accidentally linked to each other. However there are ways to mitigate these risks by automatically checking for such issues.

### Workflow

The DMF is a development environment for the various data standards that arise in the context of DiSSCo rather than documentation hub for released standards.

The general editing workflow for the creation of a new standard would be as follows: Terms that are reused from other standards are imported into the DMF using QuickStatements. New terms are either created within the Wikibase instance or imported as well, if they have been initially created elsewhere, e.g. in a spreadsheet. The terms are then further refined, edited, translated and reviewed by various people. Discussions about specific terms can be held on the wiki Talk pages associated with each item. For general discussions and documentation, e.g. about a certain group of terms, wiki pages in the Project-Namespace (i.e. wiki pages that start with Project: in the title) can be used.

During the editing process, maintenance scripts are run at regular intervals. They look for potential issues or editing mistakes, like missing connections, contradictions or links to terms that are not part of the current standard. The maintenance scripts are SPARQL queries that are run via the Query Service and ideally should return not results. If results are returned, they need to be checked and fixed manually. Not all issues that are found require fixes though, as the scripts might be “overly cautious” and report properly modelled items as problematic that can be ignored after being reviewed.

Once a consensus is reached that a specific standard has achieved a certain the maturity, a release can be made (even if it is just a release for public review, e.g. a beta version). For the release the terms are converted into versioned instance of themselves (see paragraph *Versioning*) and the release script is run (see paragraph *Release*) to export the complete standard in its desired target format, including a format that is suitable as a stable documentation for the terms.

### Versioning

Due to the inherent versioning of the wiki system, each change to a page or item is tracked and any previous version can be publicly viewed or restored. However for the context of versioning of a semantic standard this is too detailed. In order to be compatible with the versioning model proposed by TDWG (Biodiversity Information Standards, <https://www.tdwg.org/> ) in their Standard Documentation Standard (SDS<sup>6</sup>) and Vocabulary Maintenance Standard (VMS<sup>7</sup>) for

---

<sup>6</sup> Vocabulary Maintenance Specification Task Group (2017) Standards Documentation Standard (Endresen D, review manager). Biodiversity Information Standards (TDWG) <http://www.tdwg.org/standards/147>

each term, there needs to be the generic term that always represents the most recent version of the term. This term then has an attribute that points to the most recent versioned term, that includes a version number or date in its URI. When a term is modified, a new versioned term is created and the generic term then points to this new versioned term as is most recent version. The previous versioned term remains unmodified, making changes of the term transparent.

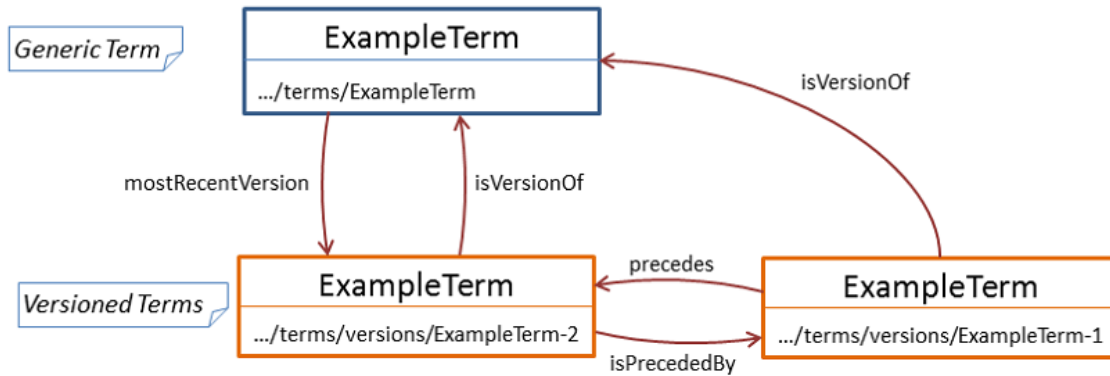


Figure 4: Schematic representation of a term, its versions and their relations

In order to represent this versioning model within the DMF, all of the item pages that are the work is done on are considered as the unversioned editions of the term. Once a release is made, a tool will duplicate the item, mark it as a versioned item, create the links between it and the unversioned item, and set the item as a protected page, so it can not be edited any further.

For this purpose a tool called Wikibase Versioning Assistant<sup>8</sup> was created for the maintenance of the ABCD Wikibase. It will also be installed for the DiSSCo Modelling Framework.

### Release

When a standard or vocabulary within the Modelling Framework is mature enough to be released, a dedicated release script is run that takes all of the terms associated with that standard and exports them into a file in the desired release format, e.g. RDF or JSON-LD, using templates. This is done by an external python script that loads the content from the Wikibase via its SPARQL Endpoint (part of Query Service suite). The generated file can then be hosted somewhere else as the released version while work on updating the standard for future versions can continue within the DMF without it effecting the released terms and their definitions. The release workflow can also be used to export the human readable documentation of the standard into other formats that can then in turn be loaded into HTML webpages that make the concepts of the standard widely accessible.

### Issue Tracking

While there are ways or concepts for handling issue tracking within a MediaWiki or even within Wikibase, it is our recommendation to not do this. Instead a dedicated issue tracking system should be used with links to the items in question. This allows for a more natural discussion and relies on the features that were built with such a use case in mind. In the case of DiSSCo we

<sup>7</sup> Vocabulary Maintenance Specification Task Group (2017) Vocabulary Maintenance Standard (Endresen D, review manager). Biodiversity Information Standards (TDWG) <http://www.tdwg.org/standards/642>

<sup>8</sup> <https://git.bgbm.org/abcd/wikibase-versioning-assistant/>

would recommend to use the Github issue pages. Each standard that is developed within the DMF should have its own Github repository and any issues or suggestions regarding the standard or even individual term, should be managed and discussed there.

## 03 General Issues and Challenges

---

### Restructuring of Wikibase Docker Container Infrastructure

At the end of June 2021 the Wikibase Docker repository<sup>9</sup> of the Wikimedia Deutschland was archived and thereby discouraging any further use of the repository to run Wikibase instances. A new installation guide has been posted with a very differently structured approach to install Wikibase as a Docker image<sup>10</sup>.

As a result the push to release the DiSSCo Modelling Framework on the production server has been halted. We believe that it is a lot less effort to restructure the setup of the DMF now and adjust it to the new Docker image without any legacy data, than to do this switch later, after the instance has been running for a while and data was already entered. The necessary adjustments however could not be done prior to the deadline of this milestone report.

Assuring compatibility with the official Wikimedia installation path is vital for future updates and compatibility with external tools and is therefore necessary.

### Visual Editor

In order to edit a page on a MediaWiki the user has to edit the page in the wiki syntax in a text area. There is a toolbar that supports the user by adding certain syntax markers, e.g. for text decorations or links. However this is still not very user friendly and the syntax is thus a bit dissuasive for new users.

A great improvement in this regard is the Visual Editor which is an extension for MediaWiki and that provides a What-You-See-Is-What-You-Get (WYSIWYG) interface for users to edit the page and instantly see how it looks, much like modern text processors on end user devices. However because of the complexity of the Mediawiki Syntax, the Visual Editor requires additional components that take care of the continuous rendering of the page.

---

<sup>9</sup> <https://github.com/wmde/wikibase-docker>

<sup>10</sup> <https://www.mediawiki.org/wiki/Wikibase/Docker>



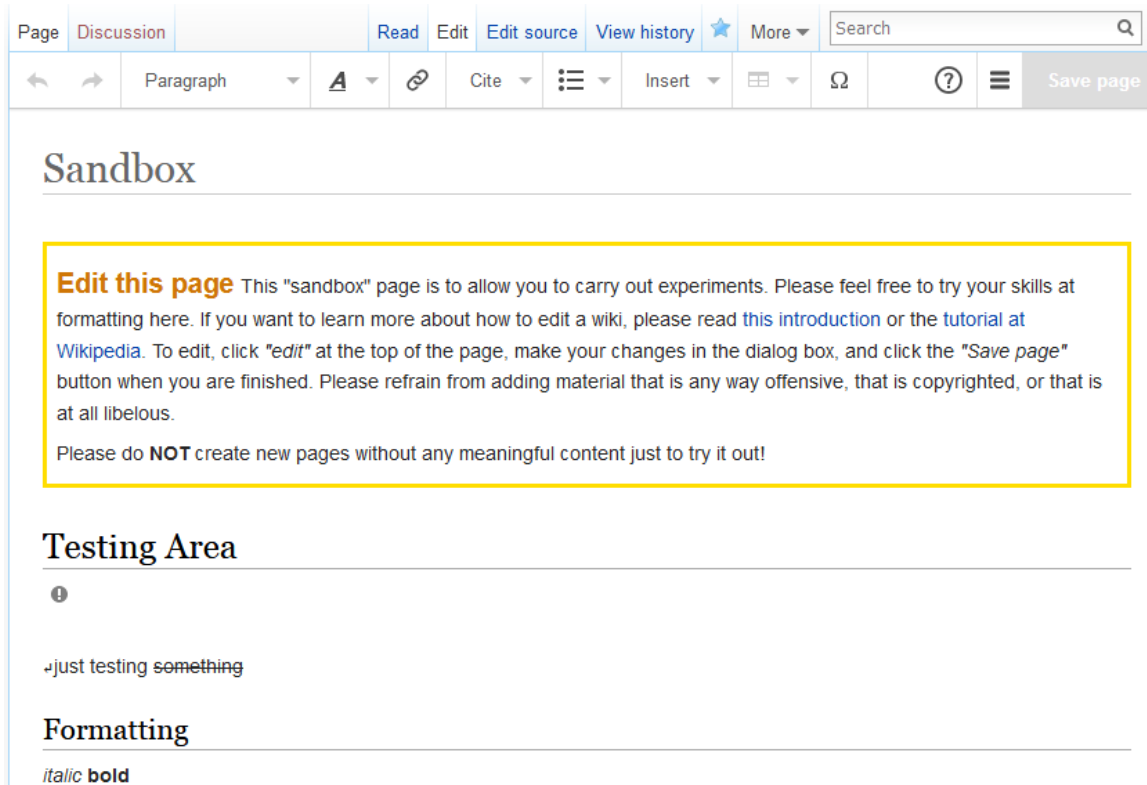


Figure 5: Screenshot of a wiki page in the Visual Editor

While most of the work with the content of the DMF will be on the Item pages provided by Wikibase, there is also the need for classic wiki pages, for instance for discussions about items, for documentation or additional content.

In the context of customizing the Wikibase Docker image we tried to install the Visual Editor as well. This however was more challenging than anticipated based on previous experiences with other Mediawiki installations. It looks like there are general incompatibilities between the Docker setup of the Wikibase installation and the requirements of the setup for the Visual Editor. We couldn't find any documentation or mention of other wikis who managed to run them together, only reports from others who run into similar problems. After several attempts we decided to not pursue this integration of the Visual Editor any further, as it is not a required for the DMF to run. Maybe the restructuring of the Wikibase Docker container infrastructure will help in this regard and later attempts will be more fruitful.

## 04 Next Steps

### Starting the Production Instance

As already outlined the next step toward the fully functional DMF is to install the customized Docker setup on the target server for the production environment. There is still some need for

clarification on minor issues, e.g. the precise setup of the server infrastructure in regard to the SSL certificate, but these are minor issues that are not considered big hurdles.

The more pressing issue is to switch the Docker setup to the new container infrastructure provided by Wikimedia. This should be done prior to the initial installation as it will make later upgrades a lot easier compared to the option to install the Wikibase now with the current setup and switch to the new infrastructure with the next upgrade with an already in use Wikibase instance.

### Integration of additional tools

Two of the tools required for the release workflow are not yet part of the Wikibase Docker setup: The Wikibase Versioning Assistant and the Export Script.

The Versioning Assistant is written as server software to be run alongside the Mediawiki Installation. It has still to be determined whether to install it separately or to integrate it into a docker container as well. Both options are feasible but need to be evaluated in terms of required effort and stability for future updates.

The Export Scripts so far a Python Scripts that run on a Jupyter Notebook of the person running the export. The goal is to move these scripts to a server environment as well, so that all users with sufficient credentials can trigger an export directly on the server.

Both of those tools however are not necessary to run the DMF in its initial state. They can be installed and configured while the DMF is already running and while users are already creating DiSSCo related standards within them, so this is not a blocker for the start of the production instance of the DiSSCo Modelling Framework.