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.

Title
DiSSCo Prepare Deliverable 5.2 "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
Sabine von Mering https://orcid.org/0000-0003-2982-7792
Marcus Ernst

Publisher
DiSSCo Prepare

Identifier of the publisher

Resource ID
https://doi.org/10.34960/e3nv-zh69

Publication year
2022

Related identifiers
https://doi.org/10.34960/wn2h-9g16

Relation type
Milestone

Is it the first time you submit this outcome?
Yes

Creation date
13/01/2022

Version
1

Citation
Fichtmueller D. & Güntsch A. (2022) DiSSCo Prepare Deliverable 5.2 "DiSSCo Modelling Framework". https://doi.org/10.34960/e3nv-zh69

Abstract
This is the deliverable report of the DiSSCo Prepare Deliverable 5.2 “DiSSCo Modelling Framework”. It describes the technical setup of the DiSSCo Modelling Framework (DMF), 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.
This report is largely based on the DiSSCo Prepare Milestone Report MS 5.6 “A functional prototype of DiSSCo Modelling Framework” (https://doi.org/10.34960/wn2h-9g16). It has been updated and expanded to include the developments, updates and changes that were done since its initial publication.

Content keywords
technical

Project reference
DiSSCo Prepare (GA-871043)

WP number
WP5

Project output
Deliverable

Deliverable/milestone number
5.2

Dissemination level
Public

Rights

Resource type
Text

Format
pdf

Funding Programme
H2020-INFRADEV-2019-2

Contact email
d.fichtmueller@bgbm.org

DiSSCo PREPARE

H2020-INFRADEV-2019-2
Grant Agreement No 871043

DiSSCo Modelling Framework

DiSSCo Prepare WP 5 – Deliverable 5.2

David Fichtmueller, Anton Güntsch

Task Lead
BGBM Berlin

Botanischer Garten Berlin

Abstract

This is the deliverable report of the DiSSCo Prepare Deliverable 5.2 “DiSSCo Modelling Framework”. It describes the technical setup of the DiSSCo Modelling Framework (DMF), 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.

This report is largely based on the DiSSCo Prepare Milestone Report MS 5.6 “A functional prototype of DiSSCo Modelling Framework”[1]. It has been updated and expanded to include the developments, updates and changes that were done since its initial publication.

Keywords

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

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

Index

01 Introduction

DiSSCo Modelling Framework

The interoperability of DiSSCo Research Infrastructure (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[2] 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. However, the editing process is quite straight forward also for novice users.

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.

[2] 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).

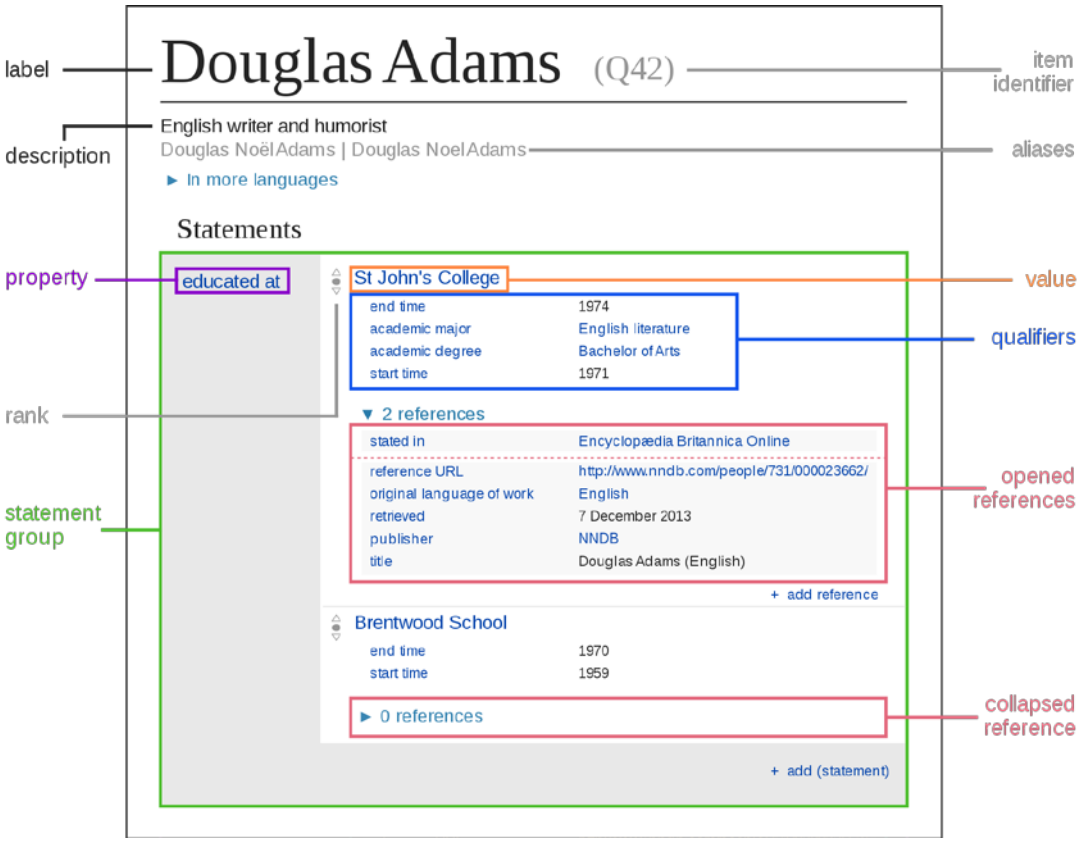

Figure 1: Diagram of a Wikidata Item
Source: Charlie Kritschmar et al., https://commons.wikimedia.org/wiki/File:Datamodel_in_Wikidata.svg

Docker

Docker[3] 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.

[3] 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[4]. Because of this we were aware that Wikibase fulfills all of the necessary requirements plus it has the benefit that the team has 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. This analysis was updated at the start of DiSSCo Prepare 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[5], 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 on the modelled topic.

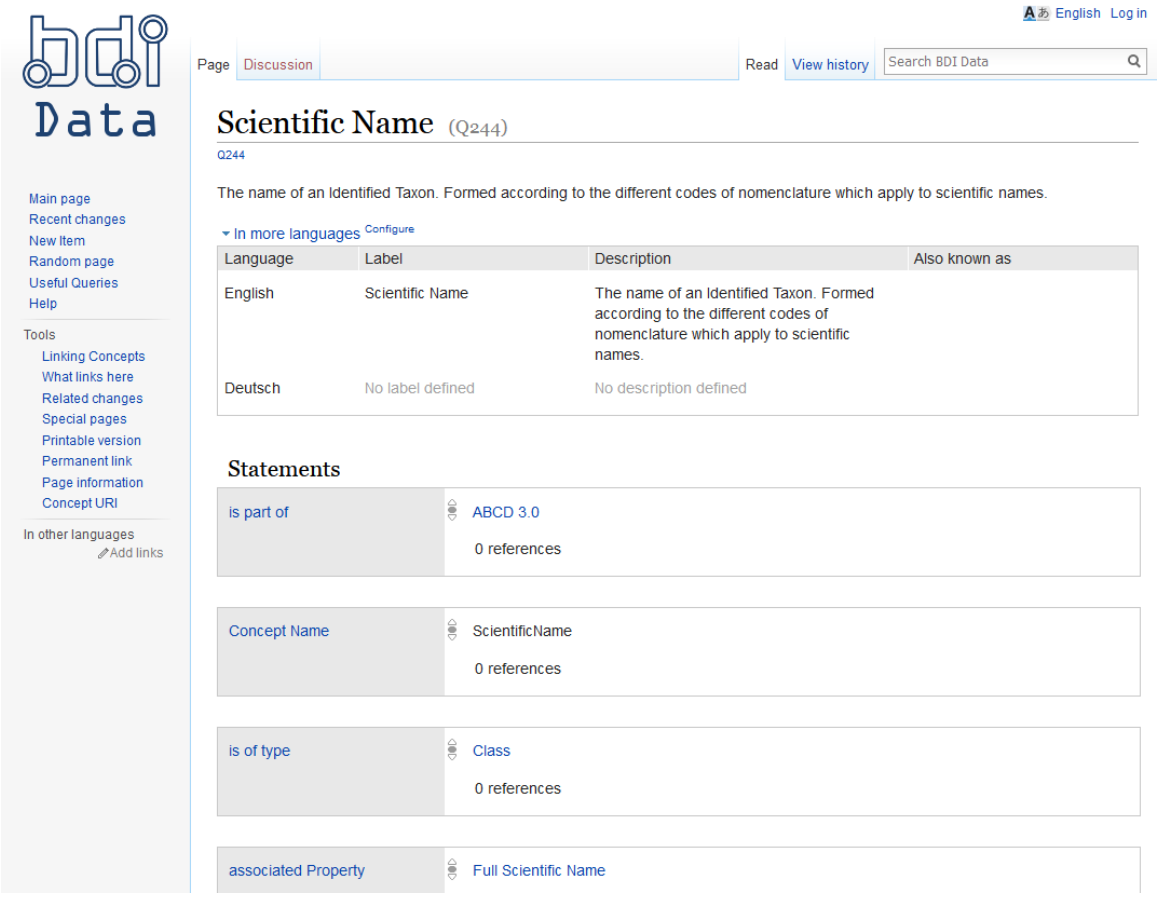

Figure 2: Screenshot of the Wikibase instance for the ABCD 3.0 Development from the BGBM, https://wiki.bgbm.org/bdidata/index.php/Q244

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

[4] 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

[5] 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[6], 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 does not 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 sits between Blazegraph and the Query Service that allows only reading access to Blazegraph, as the proxy is publicly accessible.

QuickStatements

QuickStatements[7] 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[8].

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 DMF 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 the setup. Here is an overview of the most important modifications:

[6] https://mariadb.org/ , a fork of the database system MySQL.
[7] https://github.com/magnusmanske/quickstatements/ , the Quickstatements installation for Wikidata can be found under https://quickstatements.toolforge.org/ .
[8] https://openrefine.org/

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 unique for the current installation. 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 that 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

Some additional extensions have been installed to improve the general functionality of the underlying Mediawiki such as PluggableAuth and OpenIDConnect for the SSO functionality.

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 have been imported into the DMF.

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.

On the server of the production instance the web proxy also takes care of the https encryption and for this it uses the tool Lets Encrypt[9] to generate SSL certificates for the server.

Improve QuickStatements to create and edit properties

On Wikidata, the ability to create new properties is limited to administrators who do this once a community consensus about the need for a new property has been 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

[9] https://letsencrypt.org/

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

The code for the Wikibase Docker setup is kept in two different git repositories. Some of the customizations outlined above require changes in the pipeline that is used to build the docker container, whereas some can be done by adjusting the configuration files for running the existing container. Generally speaking, the adjustments regarding the pipeline could be of interest to the wider Wikibase community and can be found in this repository:

https://github.com/DiSSCo/wikibase-release-pipeline/tree/dissco

and the changes for the DiSSCo specific configurations are available in this repository:

https://github.com/DiSSCo/modelling_framework_config

The code from both repositories is required to run the DMF with all of the adjustments in place.

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.

Prior to the *Restructuring of the Wikibase Docker Container Infrastructure* (see next sub-chapter), we reached out to the Wikidata/Wikibase development team and asked them if they would consider any of these changes to be included in the 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. We got a reply that showed general interest in some of the modifications, however, it would require the changes to be adjusted to the new release pipeline and for some of the larger features it would also require discussion with the larger Wikidata/Wikibase community. We were encouraged to create pull requests for the smaller features and general tickets for some of the larger changes, to have a central point where the community can discuss them and come to a consensus if they should be implemented based on our approach.

So far we have not done this, but the structure of the repository is set up in a way to allow for easy sharing of individual adjustments, since each of the features was initially developed in a dedicated git branch before being merged into the combined ‘dissco’ branch that is used for the deployment on our server. These development branches make it easy to create pull requests for individual adjustments, independent of each other.

Restructuring of Wikibase Docker Container Infrastructure

At the end of June 2021, the Wikibase Docker repository[10] of the Wikimedia Deutschland was archived, thus 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[11] and a new code repository was provided[12].

This required extensive adjustments of the customizations and adjustments that were done for the prototype installation. However, since the production server was not set up at that point, it provided the possibility to restructure the setup and use the new Wikibase Release Pipeline for a fresh installation. This way it was not necessary to switch the setup while the installation was already running and had gathered legacy data, which would have made up updating process a lot more complicated.

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

With the update some of the issues that we addressed with previous adjustments were also solved by the Wikimedia development team. This included the installation of many of the extensions we installed additionally (e.g. the SyntaxHighlight or ParserFunctions), moving the Mediawiki configuration into a persistent folder and moving sensitive information into a dedicated file that is not shared.

Additionally, the new release pipeline also has the VisualEditor pre-installed which was on the list of additional things to implement for the production instance. It 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. This greatly improves the usability of the wiki for edits on classic pages that are not the Wikibase Item pages, e.g. documentation and discussion pages.

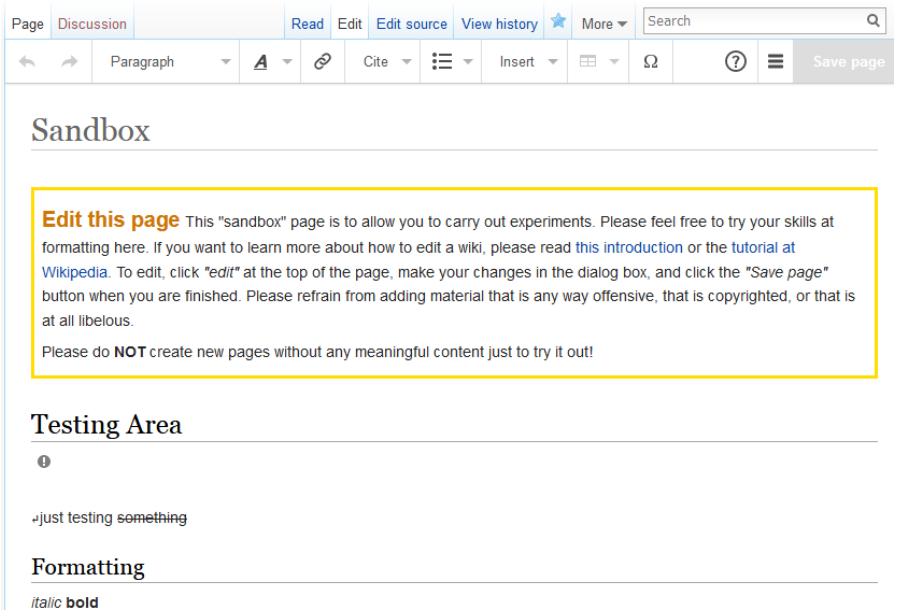

Figure 3: Screenshot of a wiki page in the Visual Editor

[10] https://github.com/wmde/wikibase-docker
[11] https://www.mediawiki.org/wiki/Wikibase/Docker
[12] https://github.com/wmde/wikibase-release-pipeline

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 exceptions of the additional web proxy and the Visual Editor, 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 prototype instance will be kept even though the production instance is already available, as a test bed and sandbox for new users who want to get used to the Wikibase environment, without affecting any official standards or documentation. However, the installation of the prototype instance is still build on the old Docker images that have been discontinued. Upgrading it will take a lot of work that is not justifiable for a test environment, so at some point in the future, it will be turned off. The need for another dedicated test environment based on the new Wikibase Release Pipeline will be evaluated again at that later point.

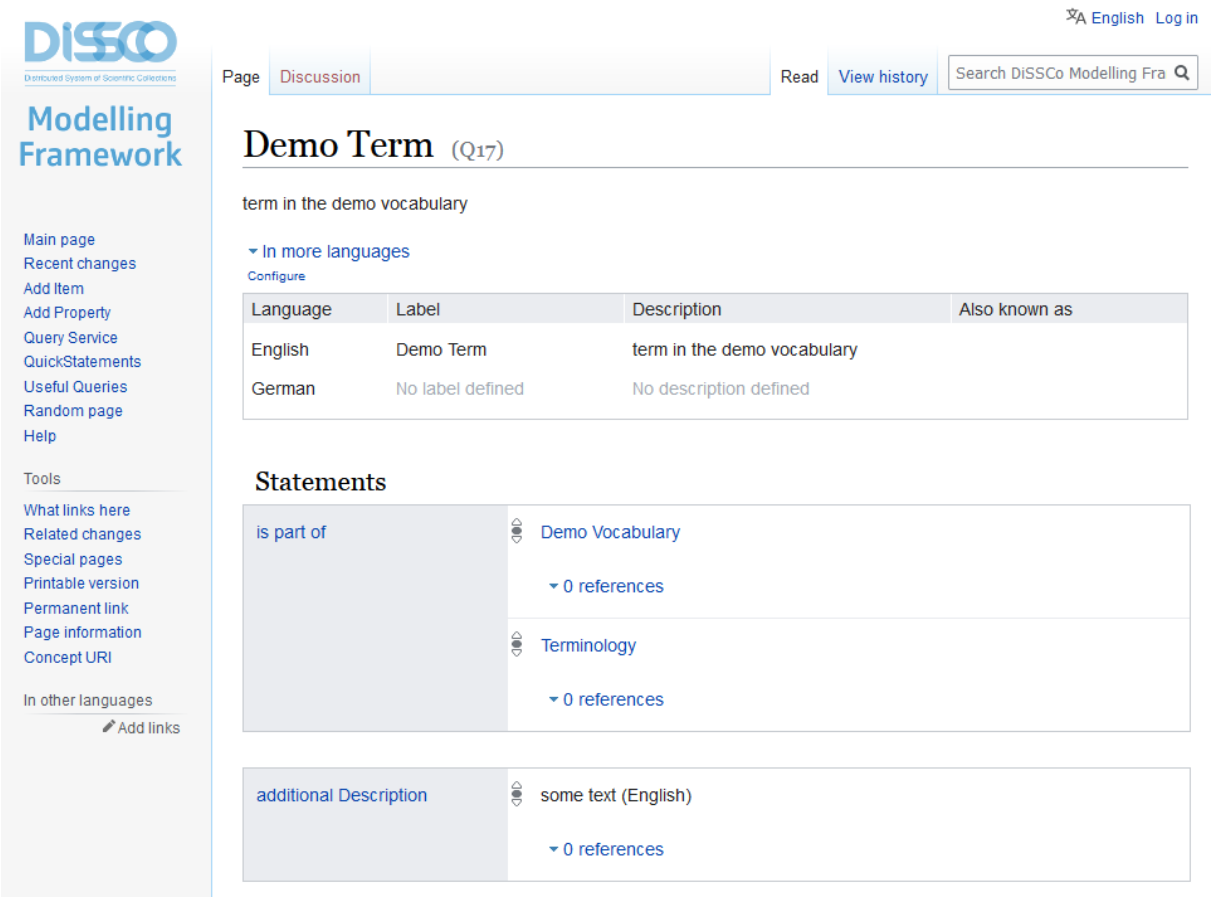

Figure 4: Screenshot of a Dummy Page on the Prototype Instance of the DiSSCo Modelling Framework: http://dissco-mf.bgbm.org:8181/wiki/Q17

Production Instance

The production release of the DiSSCo Modelling Framework is now available at https://modelling.dissco.tech/ and the related tools are available via dedicated subdirectory paths, e.g. https://modelling.dissco.tech/quickstatements/ for Quickstatements and https://modelling.dissco.tech/query/ for the Query Service.

The service runs on a server that is available through Amazon Web Service (AWS)[13] and uses the dedicated AWS email services to send emails to the user, e.g. for password resets.

There is a backup script running that does nightly backups of the Docker volumes and configurations that are running on the server. This allows restoring the data in the event of a failure or malicious user interactions to a previous state.

Conceptual Work

This section describes how the work with the DMF for the creation and release of a specific standard will be conducted. 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. 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. This might result in terms from different and supposedly independent standards accidentally being 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.

[13] https://aws.amazon.com

Once a consensus is reached that a specific standard has achieved a certain 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 a 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 of 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[14]) and Vocabulary Maintenance Standard (VMS[15]) for 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 than points to this new versioned term as is most recent version. The previous versioned term remains unmodified, making changes transparent.

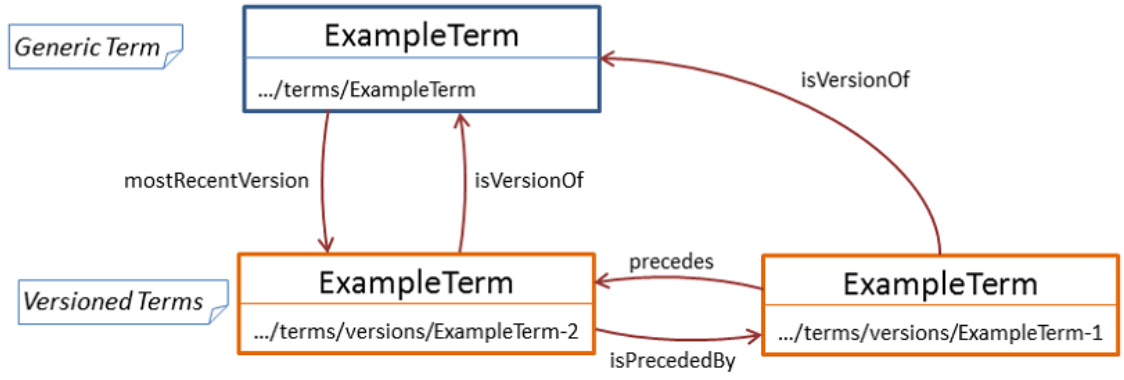

Figure 5: Schematic representation of a term, its versions and their relations

In order to represent this versioning model within the DMF, all of the item pages that are being worked 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[16] was created for the maintenance of the ABCD Wikibase. It will also be installed for the DMF.

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

[14] Vocabulary Maintenance Specification Task Group (2017) Standards Documentation Standard (Endresen D, review manager). Biodiversity Information Standards (TDWG) http://www.tdwg.org/standards/147
[15] Vocabulary Maintenance Specification Task Group (2017) Vocabulary Maintenance Standard (Endresen D, review manager). Biodiversity Information Standards (TDWG) http://www.tdwg.org/standards/642
[16] https://git.bgbm.org/abcd/wikibase-versioning-assistant/

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 tern 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 not to 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 would recommend to use the GitHub issue pages. Each standard developed within the DMF should have its own GitHub repository and any issues or suggestions regarding this standard or even individual term should be managed and discussed there.

03 General Issues and Challenges

Issues with the new Wikibase Release Pipeline

Due to the new structure of the Wikibase Release Pipeline, not all of the adjustments that were part of the previous DMF docker setup could be easily converted and integrated. This is the case for two different sections of automated content import during the installation process. As a workaround, the content was imported manually into the production installation. These changes would require some community discussion if they were to be integrated into the general Wikibase installation process. Therefore, there is no need for a git branch for a dedicated pull request.

Single Sign On

One of the desired features was to have a Single Sign On (SSO) so that users could login with other existing accounts, ideally one that they already have and use within DiSSCo. The DiSSCo Authorisation and Authentication Infrastructure (AAI)[17] would be a good solution for this, however, it is still in its being developed and not yet ready for productive use. As an alternative, we tried to use the SSO Service from ORCID[18]. While it was possible to connect to ORCID using OpenID, we encountered a strange bug. The Mediawiki installation would not accept the user name provided by ORCID and instead generated a new username “User” followed by number that increased for each new user. It was possible to rename the users to their desired user name manually. The issue might be caused by a configuration error, but it is hard to test as each attempt to evaluate changes to the configuration requires a new user account to log into Mediawiki via SSO for the first time. So in the end we decided that trying to fix this bug was not

[17] https://www.dissco.eu/services/#aai
[18] https://orcid.org

worth the expected effort and the feature provided not much additional benefit, so we turned the SSO off again for the time being. Instead, the users can now register and login via the default Mediawiki user management. Once the AAI is available, we will add it as the SSO option for the DMF.

04 Next Steps

Continuously evolving the DiSSCo Modelling Framework

While the general setup of the DiSSCo Modelling Framework is complete, we expect some need for further adjustments once it is used in production and new user needs arise. So we will continue to adjust and improve the installation of the DMF in close cooperation with the partners in Task 5.2. This also applies to updates and patches for Wikibase that come from Wikimedia, in particular security updates.

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 this Deliverable.