Project Number:	720270	Project Title:	Human Brain Project SGA1

Document Title:	D5.8.1 - Strategy and Architecture for HBP Data Viewers
Document Filename:	D5.8.1 (D73.1 D50 - SGA1 M12) ACCEPTED 180709.docx
Deliverable Number:	SGA1 D5.8.1
Deliverable Type:	Report
Work Package(s):	
Dissemination Level:	PU (= public)
Planned Delivery Date:	SGA1 M12 / 31 March 2017
Actual Delivery Date:	Submitted: 31 May 2017; Rejected: 7 Aug 2017; Resubmitted: 31 January 2018; ACCEPTED 09 Jul 2018

Authors:	Jeff MULLER (EPFL; SP5); ; Timo DICKSCHEID (FJZ; SP2, SP5)
Compiling Editors:	Roman VOLCHENKOV (UiO, SP5), Jan BJAALIE (UiO; SP5), Timo DICKSCHEID (FJZ; SP2, SP5)
Contributors:	Timo DICKSCHEID (FJZ; SP2, SP5); Jeff MULLER (EPFL; SP5); Jan BJAALIE (UiO; SP5); Dmitri DARINE UiO; (SP5); Michael DENKER (FZJ; SP5); Anna KRESHUK (UHEI; SP5); Benjamin WEYERS (RWTH; SP7); Colin MCMURTRIE (CSCS; SP7); Daniel MALLMANN (FJZ; SP5); Stefan EILEMANN (EPFL; SP7); Claudia HÄNEL (RWTH; SP7); Yann LEPRINCE (SP5); Jonathan LURIE (Evans Lab, McGill University);
SciTechCoord Review:	EPFL (P1): Jeff MULLER, Martin TELEFONT UHEI (P47): Martina SCHMALHOLZ, Sabine SCHNEIDER
Editorial Review:	EPFL (P1): Guy WILLIS, Martin O'NEILL

Abstract:	Because there are a wide range of software systems and architectures which might help users to visualise HBP-SP5 data, it was necessary to clearly establish the why, what and how of the HBP Data Viewers. The approach taken is a common one, staring with use cases, analysing requirements and then proposing an architecture which should satisfy the requirements. It is expected that this document will serve as a vital resource in SGA1 and a concrete starting point for a systematic discussion of how visualisation server neuroscience in the HBP and beyond.

Keywords:	

Table of Contents

1. Introduction ... 3
2. Considerations ... 3
3. Use cases ... 3
3.1 SP5-VA-UC01 - Analysis as a service and Embedded Visualisation Engines ... 3
3.2 SP5-VA-UC02 Large Volume and sub-volume alignment ... 4
3.3 SP5-VA-UC03a Integrated batch feature extraction and segmentation with ilastik ... 6
3.4 SP5-VA-UC03b Interactive segmentation and feature extraction in 2D and 3D ... 7
3.5 SP5-VA-UC04 Exploring human and rodent 3D atlases ... 8
4. Combined Requirements ... 10
4.1 Definitions ... 10
4.2 Essential features ... 11
4.3 Useful features ... 12
4.4 Potentially useful features ... 12
5. Architecture Component Candidates ... 12
5.1 Neuroglancer - Multiprotocol web-based viewer ... 12
5.2 OpenSeadragon ... 17
5.3 DVID - Distributed Version Image Database ... 17
6. Proposed Architecture and Roadmap ... 19
6.1 Main categories of viewers required ... 20
6.2 Phase 1 ... 24
6.3 Phase 2 ... 25
6.4 Possible Future Phases ... 26
7. Conclusions ... 28

List of Figures

Figure 1: Mock-up of the volume alignment tool interface, showing a source and target volume side by side. ... 6
Figure 2: Mock-up of the HBP 3D web-based atlas viewer ... 9
Figure 3: Distributed Version Image Database ... 17
Figure 4: Strategy and architecture for data viewers - Phase 1 ... 25
Figure 5: Strategy and architecture for data viewers - Phase 2. ... 26

List of Tables

Table 1: Formats for viewers or data services ... 11
Table 2: Feature comparison matrix of Neuroglancer and other web-based viewers. ... 14
Table 3: Viewer strategy and architecture categories ... 20
Table 4: Currently used image viewers ... 21

1. Introduction

The purpose of this document is to describe the prioritised use cases which will be targeted by the HBP Data Viewers and to define a future-proof architecture for their immediate implementation. This architecture should also have reasonable efforts made to accommodate later expansion of capabilities. The architecture will be defined in detail where it is known, with a clear identification of areas for potential future investigation.

The document starts by outlining a range of use cases which should be addressed in some way by the viewer architecture. Each use case includes its specific requirements and other considerations. This is followed by a rationalised requirements list with prioritisation for key use cases or portions of use cases. Potential candidates for integration into the system architecture are documented. Finally, the document provides an architecture based on the best candidates in the evaluated system component candidates.

2. Considerations

Basic viewers are needed for initial exploration of data, for example as a first step when rapidly inspecting data retrieved after performing a query in the KnowledgeGraph.

Advanced viewers have a range of functionalities, determined by HBP use cases and workflows. Advanced viewers usually combine several data categories. Basic viewers and advanced viewers are either built using the same tool kits or built as separate efforts. Only cost-benefit analysis is applied to evaluate viewer creation. Further, it should be clear that basic viewers are very cheap to develop; in most of the cases we just reuse existing frameworks and tools.

Choice of libraries and frameworks is not deemed critical for the development. The critical point will be documentation and clarity of the shared code, facilitating maintenance and co-developments. It is with this view that the candidates above are evaluated in the sections below.

Finally, it should be remembered that the original 2D or 3D datasets are uploaded and stored in FENIX. Viewers for these datasets will subsequently need acceleration structures either internally [i.e. the Distributed Version Image Database (DVID)] or externally (i.e. precomputed image pyramids for Neuroglancer)

The viewers then access those acceleration structures, not the original data. In the case of the precomputed image pyramids of Neuroglancer: If you zoom in to a specific area of a large image, then only those tiles (from certain level) needed to show the specific areas are downloaded (in contrast to loading the whole image). The same applies to OpenSeadragon-based viewers.

3. Use cases

3.1 SP5-VA-UC01 - Analysis as a service and Embedded Visualisation Engines

3.1.1 Actors

Anna - an image processing expert; Benni - a microscopy expert producing volumetric data.

3.1.2 Success Scenario

1) Anna receives from Benni a link to a Tier0-curated volumetric dataset that he registered to the Neuroinformatics Platform (NIP). The NIP dataset card that she finds there

contains a FENIX or Collab Storage URL pointing to the corresponding NIFTI file, which is 300GB in size. Anna now wants to analyse the data using Jupyter notebooks.

2) Anna reviews the metadata stored in the NIP, to ensure that the data was gathered ethically and matches the requirements for her experiment.
3) Using the Storage URL, Anna can access the file from within a Jupyter notebook hosted by the Collaboratory, and runs the data file through Numpy to gather statistics and create visual plots.
4) Anna runs the data through an image filter to produce a new volume (also 300GB).
5) To inspect the filtered data, Anna uses notebook-embedded python code to load this data into a 3D volumetric viewer embedded in the Jupyter notebook. The visual presentation and navigation in the 3D view matches to the one used by the HBP 3D atlas viewers.
6) She shares this notebook with Benni. Benni can see the same visualisations, and hereby verify that the analysis makes sense.
7) Anna releases the processed data set to a public viewer instance. Here it can be viewed through the internet by members of her image processing MOOC using their web browsers.
8) Catherine, a MOOC student, loads up a public Collab page which contains the viewer which Anna embedded and can view the processed volume overlaid with a standard reference atlas.

3.1.3 Requirements

Due to divergent requirements on the consumer audience, it's best to separate this into two scenarios and consider each separately. One scenario involves small group collaboration and the other involves a large group audience.

SP5-VA-UC01a: Small group collaboration:

1) Users: 2-10 at a time interacting with a single dataset.
2) Update frequency: 5-20 updates/hour during heavy update cycles.
3) Data lifecycle: Variable.
4) Some of the intermediate results would need to be tagged for long term storage.
5) Some would likely be uninteresting and could be deleted.
6) Data volumes: additional data volumes may be processed along with the original volume. Volumes processed per update iteration could be as high as 1TB (3x300GB)/update.

SP5-VA-UC01b: Large group audience:

1) Users: 20-300 at a time interacting with a single dataset.
2) Update frequency: 1-2 updates/month during course preparation.
3) Data lifecycle: Final results.
4) Most results which were delivered to this group would need to be tagged for long term storage.
5) Data volumes: additional data volumes may be visualised along with the original volume. Volumes *accessed* per view could be as high as 1TB (3x300GB)/view. No use would need to view all data interactively so special care would need to be taken to deliver only a small subset of the interesting data.

3.2 SP5-VA-UC02 Large Volume and sub-volume alignment

3.2.1 Actors

Alex, a Post-Doc working on high-resolution histology.

3.2.2 Success Scenario

1) Alex uploads a high-resolution 3D reconstruction of a local brain region in the Collab, which is represented as a NIfTI-2 file in the order of several hundreds of Gigabytes.
2) To make his datasets accessible for online atlas tools, Alex visits the Dataset registration form:
   a) Alex fills in minimal metadata
   b) Metadata enters the KnowledgeGraph
   c) Image data enters an instance of DVID
3) Alex views the dataset card, and clicks on a link to the volume alignment tool.
4) Before entering the tool, Alex is asked to select an HBP- supported reference template from a list.
5) Alex is then automatically directed to a web-based partial alignment tool, which shows a triplanar view of both the reference volume (target) and Alex' new volumetric dataset (source). The two views are displayed side-by-side (Figure 1). The appearance and navigation of the triplanar views matches that of the HBP 3D atlas viewer closely, and is therefore intuitive to Alex.
6) Alex adds corresponding landmarks between the source and target volumes (Figure 1).
7) Based on the corresponding landmarks, a spatial transformation is computed on Alex's request (affine, 4×4 matrix in projective geometry), and applied to warp the source data volume closer to the target volume.
8) Alex can add, remove, or update landmarks, and iteratively refine the spatial transformation by cycling through steps 6 and 7 repeatedly.
9) After being satisfied with the spatial alignment, Alex can store the resulting transformation parameters it into the KnowledgeGraph where it becomes accessible to the rest of the HBP tools.

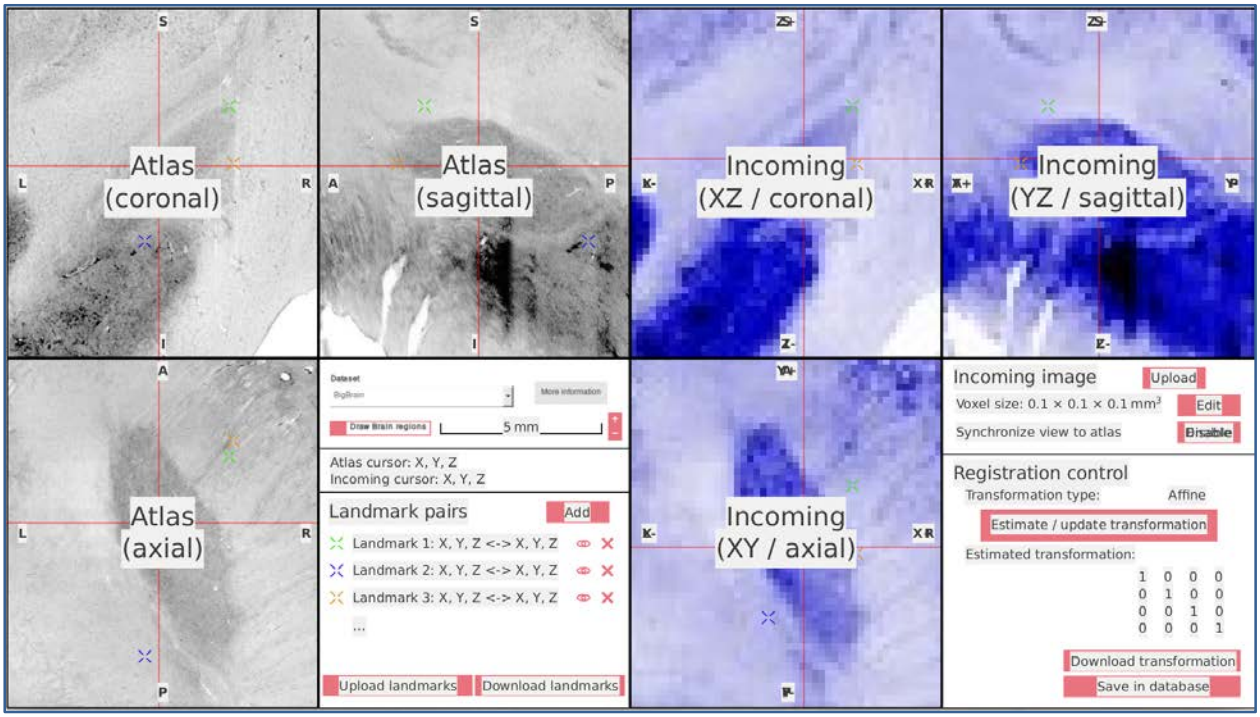

Figure 1: Mock-up of the volume alignment tool interface, showing a source and target volume side by side.

The tool allows to iteratively enter corresponding 3D landmarks to compute and refine a 3D spatial transformation between the datasets.

3.2.3 Requirements

1) An interface for the user to upload an image and its metadata, and get it entered into DVID so that it can be displayed by the tool. A range of input formats should be supported, initially Nifti-1, Nifti-2 and HDF5 should be included.
2) For the incoming dataset: 2D image tiles (greyscale or RGB) are computed and served by DVID
3) Tiles of the transformed image are served by DVID. Two approaches are possible:
   a) (preferred) DVID can re-sample the image on the fly, with an affine transformation passed as URL parameters (partially implemented for the `imageblk` datatype).
   b) (fall-back) the tool can re-sample the image on the server side, creating a transformed NIFTI file, which is then entered into DVID. This approach is less preferable because the user must wait between for the whole re-sampling to be performed before viewing the result.
4) For the often-used, large template images: 2D multi-resolution image tiles are pre-computed and served statically over HTTP (0–1TB per template, less than 1–2 updates/year).
5) Infrastructure must support approximately 10 simultaneous users.

3.3 SP5-VA-UC03a Integrated batch feature extraction and segmentation with ilastik

Actors

Ada, a researcher who needs to process large images; Billy, a student.

Success Scenario

1) Ada wants to analyse a large brain dataset she found in the Collab.

2) Billy, one of her students, has the task of training a pixel classifier to extract features in this dataset using the ilastik pixel classification workflow. Billy either downloads a smaller sub-volume or accesses the dataset directly from his local ilastik installation. After Billy has finished training the classifier, he saves the ilastik project to an h5-file and uploads it to the Collab for Ada to review. The project metadata contains links to the dataset.
3) Ada opens her dataset in the HBP web viewer and selects a representative field of view.
4) She presses the "Process with ilastik" button in the viewer. A pop-up opens that lets Ada select the appropriate ilastik-project containing the classifier. She confirms the selected project and an ilastik process is started in headless mode on the server that processes the current field of view. The result of the processing is a map of label probabilities for every visible pixel.
5) After closing the pop-up, Ada can monitor the processing progress with a progress-bar that appears in the viewer. Once the processing is finished, the progress-bar colour is changed to green and it becomes clickable. Upon clicking, the additional image layers (class probabilities) are loaded into the viewer.
6) Ada reviews the pixel classification results; in the web viewer she can see the original image data overlaid with label-probabilities.
7) Ada is satisfied with the classification result on the data she is viewing. She presses another button to start large-scale ilastik processing. She needs to select where the results will be saved to and how much resources she wants to consume (how many cores to book, how much RAM she can take on each core).
8) While she can keep watching the progress from the viewer, closing the viewer does not terminate the processing on the server. The status of the computation can also be observed in the Collab. The computation can be stopped at any time, with an option to delete the intermediate results.
9) Link to Billy's project is stored in the metadata of the results dataset.

Requirements:

1) A web-based viewer for the display of multiple large image (2D, 3D) datasets (raw data, classification results…). It must be possible to overlay the different datasets in layers and adjust transparencies and colour tables of the respective layers independently.
2) User controls connected aware of the web-viewer context (opened dataset(s)) have to be provided that allow for processing the field of view with a selected ilastik project as well as processing the whole volume.
3) The web-viewer context must be able to provide its viewing frustum to other browser code via a direct JS API call or a low-latency server-side intermediary.
4) Event mechanisms (REST API) need to be implemented that allow for monitoring progress of processing and trigger events, e.g. loading the processed data into the viewer, upon its completion.
5) Indicators of processing progress and status (success, failed, with warning…).
6) Possibility to book computational resources and save potentially large result files, with all appropriate metadata and KnowledgeGraph registration
7) (optional) Possibility to save snapshots of the current view with all the overlays at the current transparency level

3.4 SP5-VA-UC03b Interactive segmentation and feature extraction in 2D and 3D

Ada, a researcher who needs to process large images; Billy, another researcher.

Success Scenario

1) Ada finds an interesting dataset in the Collab.
2) She opens the dataset in the web viewer and wants to extract its features with ilastik.
3) She creates the appropriate number of labels and proceeds to label pixels with a brushing tool. An eraser is available for the inevitable mistakes.
4) After adding at least one labelled pixel for each class and selecting the filters to be used as classification features, she presses the "Update" button.
5) On the server, the ilastik service trains a classifier with the provided labels and features.
6) The service signals that it has finished and the viewer requests current classifier predictions for the current field of view.
7) Ada reviews the predictions, changes her filter selection or adds more labels at the locations where the predictions are wrong. She presses the "Update" button again.
8) Repeat steps 5-7 until Ada is satisfied with the predictions.
9) Ada presses a button to process the whole dataset. From this point on, the scenario follows points 7-9 of SP5-VA-UC03a.
10) Ada saves the project into the Collab. If she saves the derived datasets, their metadata contains a link to the project.
11) Billy finds another dataset with similar characteristics.
12) He opens Ada's project, adds his dataset to it and continues the labelling, finally saving his updated project under a different name.

Requirements:

- 2-way communication between the viewer and the ilastik service
  - Invalidating the outdated viewer cache after classifier or parameter update
  - Displaying the results as they come (the backend is computing block-wise), in the corresponding overlay(s)
- Annotation support in the viewer, for displaying and transmitting to the backend:
  - Pixel-level: brushing, erasing
  - Object-level: clicking, selection
- Import and export of annotations.
- If the viewer supports a multi-scale pyramid, the back-end can serve down sampled results, but the actual computation always happens at full resolution..

3.5 SP5-VA-UC04 Exploring human and rodent 3D atlases

3.5.1 Actors

Markus and Jeff, both neuroscientists

3.5.2 Success Scenario

1) Markus is planning to run a data analysis over a range of neuroimaging data, and looking for a high-quality human brain parcellation that meets his needs.
2) He goes to the HBP website to look for help, finds a prominent link to the HBP human and rodent atlas categories, and therein a range of recommended atlases.

3) Markus is attracted by the JuBrain atlas, and follows the website's recommendation to explore it in the 3D atlas viewer. By clicking on the link, he is presented an interactive 3D view of the MNI Colin 27 template, overlaid with numerous areas of the JuBrain cytoarchitectonic atlas (Figure 2).

4) The atlas viewer also displays a hierarchical list of the JuBrain atlas nomenclature, from which Markus selects a different area. The 3D viewer automatically changes the viewpoint so that this area is highlighted.

5) Markus likes this dataset, and uses his web browser to bookmark the page he is looking at. He wants to tell his colleague Jeff to have a look at this particular area, so he also sends the URL to Jeff by email. When Jeff clicks on the URL, he sees the exact same view onto the template and parcellation as Markus.

6) Besides inspecting the atlas in 3D, the viewer provides menus to select other HBP-supported template spaces for rodent and human. Markus selects a different template, which is then displayed. He realises that the list of parcellations is updated automatically. After choosing a parcellation, the corresponding hierarchy of brain regions is also refreshed.

7) Markus decides that the JuBrain atlas is the best choice for his project. He uses his browser's bookmark to go back to the JuBrain view. In the list of parcellations, he finds a link that directly brings him to a NIP website showing the dataset card of the JuBrain atlas. Here he reads all relevant metadata, and finds a link to download the most recent version to his computer.

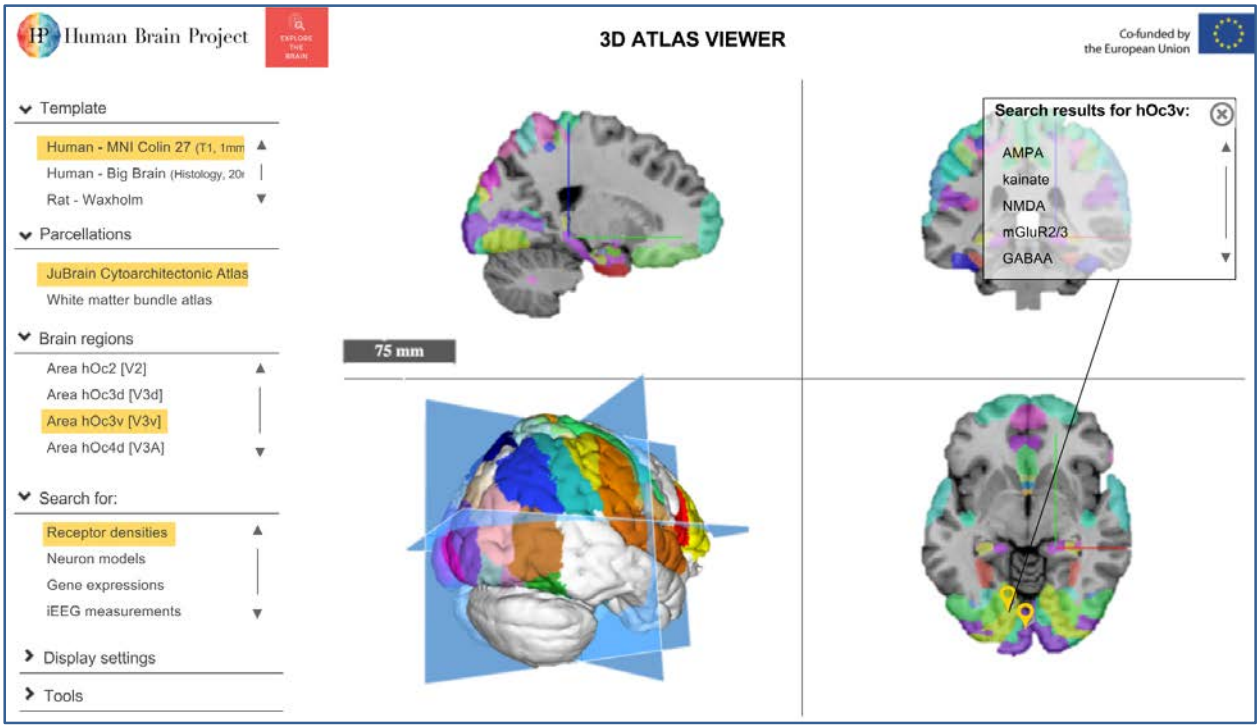

Figure 2: Mock-up of the HBP 3D web-based atlas viewer.

The central element is an instance of a triplanar volumetric viewer, together with a surface-based rotatable 3D overview. The user can choose different template spaces, select from a list of parcellations that correspond to a template space, and navigate a list of brain regions that correspond to the selected parcellation.

3.5.3 Requirements

- A software component for web-based remote display of large volumetric datasets. The view needs to display a triplanar view as well as a surface-based rotatable 3D overview, and must be able to transparently overlay coloured parcellations/maps. It must provide

an API to accept control events, such as adjusting the view to display a specified 3D coordinate. It must be compatible with the HBP image service standards.

- A dynamic hierarchical menu system to display template spaces, parcellations, and brain regions according to the HBP atlas nomenclatures. Selection events in the menu need to trigger API calls to the embedded viewer component.
- A clear URL scheme for storing persistent display settings and data view parameters. This URL scheme must make it possible to share a view by sharing a URL.

4. Combined Requirements

The following requirements are predicated on a definition for an “atlas”. Previously this has been loosely and inconsistently defined. To discuss this in concrete terms it is necessary to define both “reference atlas” and “atlas” in detail.

4.1 Definitions

4.1.1 Reference atlas

This definition is largely based on the practices of the Allen Institute in their provision of the Allen Brain Atlases.

Such a reference atlas contains:

1) Region-hierarchy.json - a tree of regions and sub-regions with an id space. The id space is shared with the annotation volume below.
2) Annotation volume - unsigned 32bit integer id voxels -> links back to ids in the region hierarchy.
3) Base imagery voxels - 16 bit grayscale voxels should be the same or higher resolution than the annotation volume.
4) Coordinate space specification:
    a) Spatial volume of a pixel
    b) Origin
    c) Landmarks of interest
    d) Resolution/precision
5) Mesh hierarchy (optional).

4.1.2 Atlas

An atlas has fewer requirements:

1) Base imagery voxels -> 16 bit grayscale voxels should be the same or higher resolution than the annotation volume.
2) One of:
    a) Coordinate space specification
        Spatial volume of a pixel
        Origin
        Landmarks of interest
        Resolution/precision
    b) Link to reference atlas coordinate space specification

4.2 Essential features

Requirements and evaluation criteria for the SP5 Image Service development in SGA1

Must have (SGA1):

1) HTTP delivery of 2D multi-resolution tiles - multiple data types must be supported.
2) HTTP delivery of 3D multi-resolution blocks - multiple data types must be supported.
3) HTTP delivery of 1D data - multiple data types must be supported.
4) Anisotropic volume support - impacts volumetric metadata requirements (see metadata "should haves" below).
5) REST API.
6) Needs to be deployed/deployable to FENIX partner sites.
7) Supports web visualisation of 10s of terabytes for 50-100 concurrent users for released atlases (reference or otherwise):
   a) Option for anonymous access
   b) Option for access control at the Collab or HPAC level
8) Supports interactive browsing and batch processing of 1-10 TBs for 2-10 concurrent users:
   a) Access control at the Collab or HPAC level
9) Data delivery services should provide strong evidence of scalable cluster IO for batch processing:
   a) 50 MB/s from a single client thread
   b) Evidence of scalability sufficient for 250 MB/s aggregated across ~10 client threads (this might be a "should have")
10) Provision of data to be delivered by the image service:
    a) For data uploaded at the time of registration:

       Unique Identifier must be created.

    b) For data in FENIX, prior to registration:

       Unique Identifier must be created.
11) Software and service components should have well defined responsibilities and APIs to allow mixing and matching of various components to best satisfy evolving requirements in current and future use cases.
12) Image viewers support:
    a) Data delivery services must support at least one web-based viewer.
13) Arbitrary slicing for multi-resolution tiles
14) A dynamic hierarchical menu system to display template spaces, parcellations, and brain regions according to the HBP atlas nomenclatures. Selection events in the menu need to trigger API calls to the embedded viewer component.
15) A clear URL scheme for storing persistent display settings and data view parameters. This URL scheme must make it possible to share a view by sharing a URL.

In addition, it is expected that the following formats should be supported by any of the viewers or data services used in the architecture.

Table 1: Formats for viewers or data services

	Raster	Vector
1D	?	?
2D	TIFF*, JPEG*, PNG*	SVG
3D	NIFTI*	STL, OBJ, X3D, NG meshes
*Requires pyramid pre-processing before viewing.		

4.3 Useful features

Should have (in SGA1 and beyond):

1) 2D stacks filmstrip viewing capabilities.
2) API availability of viewer frustum information and other view metadata.
3) Data delivery services should provide strong evidence of scalable cluster IO for batch processing with evidence of scalability sufficient for 250 MB/s aggregated across ~10 client threads.
4) 10GB should be <=20 s to ingest into image service.
5) API for delivery of mesh or geometric data in the atlas space.
   a) Separately, there is a need for a workflow for generating these meshes.
6) 1TB volume should take <=12 h to ingest.
7) Stores volumetric metadata.
8) Support from big data analysis frameworks based on Spark.
9) HTTP Delivery of 2D+t multi-resolution tiles - multiple data types must be supported.
10) HTTP Delivery of 3D+t multi-resolution tiles - multiple data types must be supported.

4.4 Potentially useful features

Nice to have:

1) Caching hints in the service API - primarily for interactive use cases or iterative analysis use cases.
2) Alternatives to HTTP in supported data delivery protocols.

5. Architecture Component Candidates

5.1 Neuroglancer - Multiprotocol web-based viewer

Neuroglancer is a non-official Google project for biological image visualisation. It supports a range of important features.

5.1.1 Main features

- Load and display volumetric data from different image services or web-hosted file formats, and extract orthogonal 2D images.
- Efficiently and interactively perform oblique projections (non-orthogonal cuts through the displayed volume).
- Loads and display 3D surface meshes.
- Display both voxel and surface data within the same space coordinate system.
- Show semi-transparent layers of multiple 3D volumetric datasets, specifically grey value and indexed datasets (e.g. template volume + voxel parcellation).

5.1.2 General architecture

Neuroglancer is a purely client-side application. It is developed in Typescript that compiles into JavaScript. Typescript allows the use of multiple new structures compared to standard JavaScript, and makes a modular web app design easier to develop thanks to strong object oriented capabilities. Typescript makes it easier to use concepts like inheritance and interface. The data must be accessible via HTTP through one of several supported services (see below). Neuroglancer does not support displaying local files on the user's computer. It relies on cross origin resource sharing ("CORS"), so the data server must explicitly allow CORS in its configuration. If the project is hosted on an *http*, it can load *https* data only if the related security settings are disabled on the web browser. Even though the project is fully client side, its architecture is composed of two modules, launched in two independent WebWorkers threads. One thread deals with user action and rendering, while the other performs queuing, downloading and data pre-processing before being sent to rendering. Having two independent threads allows having a responsive UI, even under high data processing and loading. Since there is no shared memory between threads or sync system in JavaScript, data are transferred from a thread to another using typed arrays. To compute oblique slices, Neuroglancer uses the concept of a 3D textures. Since there is no native support of 3D textures in OpenGL ES/WebGL, it simulates this concept with WebGL GLSL.

5.1.3 Supported data sources

A range of different image services and formats are supported, most importantly:

1) ***NDstore/Open Connectome.*** Neurodata NDstore provides a scalable database cluster for the spatial analysis and annotation of high-throughput brain imaging data called Neurodata Web Services. It opens the door to have applications use Neuroglancer as the foundation of their viewer to overlay Human Connectome Project data repositories.
2) ***DVID.*** DVID stands for Distributed, Versioned, Image-oriented Dataservice. It was developed by Janelia Research Campus (VA, USA). DVID is currently used in production research at Janelia and is suitable for production use for other labs as well. It is a candidate for the HBP image services and described in detail further below.
3) ***Web-hosted precomputed multi-resolution formats.*** Neuroglancer's internal chunked data structure can be stored on disk directly, in the form of JPEG files (one file per 3D chunk, where the third dimension is stored as a concatenation of 2D images along the vertical axis). This can then be accessed by Neuroglancer using the `precomputed://` protocol when hosted on a publicly accessible web server. This format is amenable to highly scalable distribution of data for datasets which don't change frequently, i.e. reference atlas releases.

5.1.4 Internal data representation

Independently of the data source, Neuroglancer arranges the voxels into chunks of data that follow a regular 3D grid pattern. *Chunk* is the unit at which the whole volume is retrieved, queued, transcoded (if necessary), copied to the GPU, and rendered. Octrees (3D quad trees)

are used to spatially arrange the chunks efficiently. All the chunks are the same size, except for the upper bound in each dimension. Chunks of different spatial resolutions can be used to represent the same image volume (multi-scale representation). The most appropriate scale is chosen on the fly, depending on the current zoom level. The chunks' size is the result of multiple trade-offs. The main ones are:

- A chunk has to be big enough to provide a large amount of data per second, knowing that the number of concomitant HTTP requests is limited in most web browsers. The bigger the chunks, the lower the number of chunks.
- It has to be small enough to allow a fast loading. Since only a small portion of each chunk will be crossed by the plane, a large portion of it will be loaded but never used.

In practice, a 64x64x64 seems to be a good compromise.

5.1.5 Data compression

The data chunks are usually encoded in uint32 or uint64. Displaying a slice full screen on a full HD monitor involves a large amount a chunks, especially if the slice is oblique and that the standard three orthogonal projections are displayed. Since caching 1GiB in CPU and GPU memory is not possible, Neuroglancer embeds a custom developed random access compression process that guarantees the spatial continuity of the whole volume. The compression processing takes advantage of the data type (uint) and the diversity of values within each block. Depending on the diversity, each block is re-encoded using a possibly smaller amount of bits per value. This number of bit per value is constant within a block but varies from one block to another. Using only native types, the compressed chunks are easily readable by GLSL (OpenGL shader language) and are faster to cache in memory.

5.1.6 Assessment

JUELICH has converted the full-resolution BigBrain (2015 release) to the "precomputed" chunk format, and deployed a prototype of Neuroglancer serving the BigBrain. The performance of this installation was promising. An obvious drawback is the lack of support for loading local files in addition to distant HTTP. The user experience is not optimal - shortcuts and controls are rather difficult to learn. At this time, we also compare Neuroglancer against other web-based viewer solutions (Table 2).

Table 2: Feature comparison matrix of Neuroglancer and other web-based viewers.

Feature	Importance of the feature for a 3D atlas viewer	Neuroglancer	Shadernavigator	OpenSeadragon	Papaya
Ability to display cross-sectional views (obliques)	absolutely required	yes	yes	n/a	no
Ability to display images larger than client-side working memory	absolutely required	yes	yes	yes	no
Ability to display volumetric data	absolutely required	yes	yes	no	yes
Display of 3D surface meshes	absolutely required	yes	no	n/a	yes

Feature	Importance of the feature for a 3D atlas viewer	Neuroglancer	Shadernavigator	OpenSeadragon	Papaya
display of indexed images (label images)	absolutely required	yes	?	yes	yes
Displays both voxel and surface data within the same space coordinate system	absolutely required	yes	no	n/a	yes
overlay intensity view with labelled image view with custom transparency	absolutely required	yes	?	yes	yes
Supports DVID image service	absolutely required	yes	no	no	no
Support Nifti display via http	absolutely required	yes	?	n/a	?
Community uptake	high	Active adoption by several communities in the field: The Openconnectome project (neurodata.io) has forked it. The EM / fruitfly community is using it. The developers of ilastik mentioned that it would be their favorite choice to comply with.	None yet. The software is in a young development stage, and not yet very stable under rotations / oblique slicing.	High in the neuroanatomy community. Several linked software projects, e.g. microdraw.	Very high in the neuroimaging community (MRI, DTI)
Efficiency displaying cross-sectional views (obliques)	high	high	potentially high	n/a	n/a
Efficiency loading canonical 2D views	high	high	potentially high	n/a	high

Feature	Importance of the feature for a 3D atlas viewer	Neuroglancer	Shadernavigator	OpenSeadragon	Papaya
Efficiency when displaying large datasets at a coarse scale	high	High when using the precomputed tile backend. Possibly poor depending on the use of other backends.	potentially high	high	poor
Efficiency when only displaying 2D views	high	moderate, if adjusting chunk sizes to this use case (allows individual setting of anisotropic chunk sizes per slice view)	?	nearly optimal	poor
Support Gifti mesh format	yes	no	no	n/a	yes
Support Nifti display for local files	yes	no	?	n/a	yes
Support Openconnectome ndimage service	yes	yes	no	no	no
Supports multi-resolution mesh	moderate	no	no	n/a	no
Support Nifti display for backend-stored files	moderate	yes	?	n/a	yes
supports time series for 2D data	moderate	no	no	no	no
supports time series for 3D data	moderate	no	no	no	no
Supports URL configuration for data access from any http service	moderate	?	?	yes	no

5.2 OpenSeadragon

5.2.1 Main features

1) Optimised 2D user experience: navigating 2D viewer is easy and intuitive – compared to 3D Neuroglancer-like applications.
2) Supported by many existing tools and applications: the formatting for tiles (as JPEGs or PNGs), e.g. DeepZoom, is widely available, non-domain specific, and easily readable.
3) Low-cost new tools development using OpenSeadragon: OpenSeadragon is an open-source, web-based JavaScript library for high-resolution zoomable images. The viewer for p.2 is available out of the box. The library has a list of plugins, e.g. scale-bar, SVG overlays, and filters, runs on both desktop and mobile, and does not require WebGL (unlike Neuroglancer).

5.2.2 Assessment

While Neuroglancer theoretically can handle 2D cases, its user interface is not designed for filmstrip viewing or for the viewing of yet-to-be-aligned image stack. Given the low cost and potentially significant usability improvements from supporting a high quality 2D viewer, OpenSeadragon should be considered for integration alongside Neuroglancer. The user experience between the two should then be streamlined, so that they use the same mouse and keyboard settings, as well as visual metaphors.

5.3 DVID - Distributed Version Image Database

DVID is a distributed, versioned, image-oriented dataservice written to support Janelia Farm Research Center's brain imaging, analysis and visualisation efforts. See Figure 3.

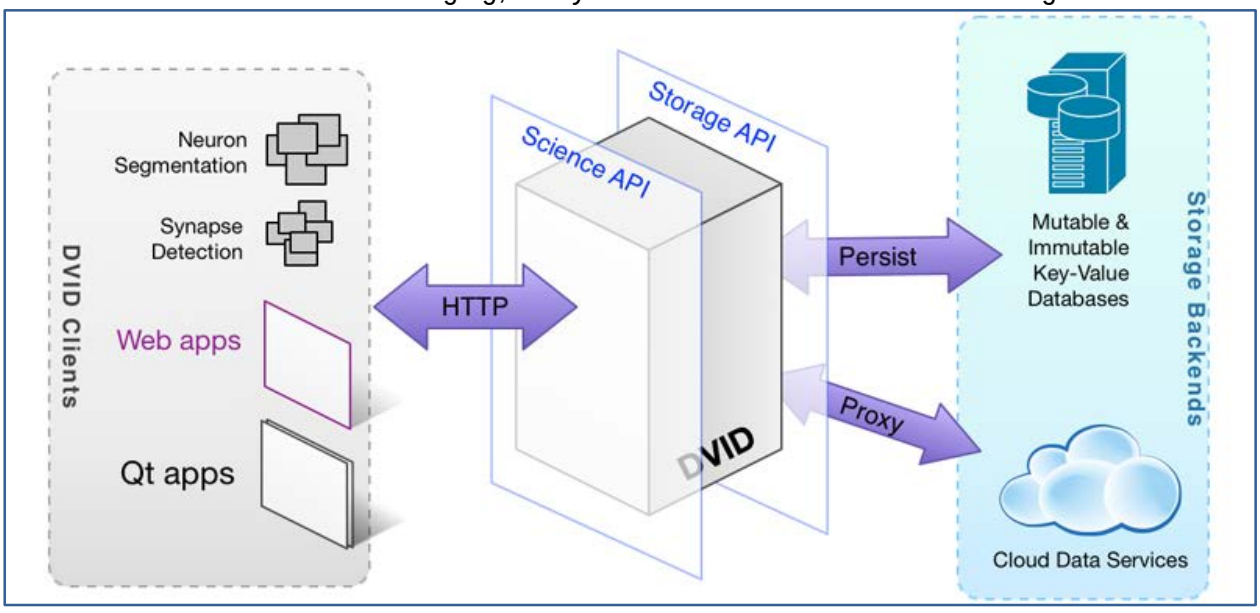

Figure 3: Distributed Version Image Database

5.3.1 Main features

- Easily extensible data types that allow tailoring of access speeds, storage space, and APIs.
- The ability to use a variety of storage systems by either creating a data type for that system or using a storage engine, currently limited to ordered key/value databases.
- A framework for thinking of distribution and versioning of data similar to distributed version control systems like git.

- A stable science-driven API that can be implemented either by native DVID data types and storage engines or by proxying to other connectomics services like Google BrainMaps or OpenConnectome.
- *Note: This means that, in principle, other analytics systems could support the DVID API for delivery of the same sorts of type extensible datasets, but under different analysis regimens.*

Pros:

- "Wide" developer community.
- Good integration with other tools:
  - Neuroglancer
  - Ilastik
- Deeper evaluation already partially done by HBP developers.
- Version control.
- Type independent.
- Existing web UI is a decent starting point.

Cons:

- Medium maturity levels.
- Deeper performance evaluations with very large datasets will be needed.
- Access control and authentication is an afterthought.
- Handling of human data in a sufficiently secure manner is an outstanding integration problem.

5.3.2 Assessment

Experience of the ilastik team:

We implemented support for reading DVID volumes primarily to support our collaboration with DVID's designers, the FlyEM project at Janelia Research Campus. Using ilastik, one can browse a (potentially very large) volume in DVID, and (for example) train a pixel classifier on remote data. Ilastik has also proven to be a valuable tool for debugging external DVID-based tools and workflows. The ilastik/DVID combination works well.

DVID exposes a simple REST API for retrieving 3D voxel data from a DVID data store. The API is simple enough that communicating with DVID can be implemented relatively easily using plain GET/POST commands via the popular 'requests' python module. But as a convenience, the DVID project also offers the 'libdvid' package, which encapsulates DVID REST API calls (and the necessary boilerplate code) into a C++/Python library. Ilastik uses libdvid for communicating with DVID. Eventually, libdvid will incorporate optimised versions of the API calls, such as enabling compression and/or raw block data transfers. Since ilastik already uses libdvid, it will be easy for us to upgrade to these API enhancements. Besides merely storing large volumes, DVID is designed with features for collaborative volume editing. These features include git-like versioning of image data, and metadata storage via access to DVID's underlying key/value database. These features comprise a large portion of DVID's design and internal complexity, but ilastik has no direct need for such features and therefore does not directly use them. These features could, however, be useful for storage of metadata necessary for the KnowledgeGraph. Another feature that ilastik does not exploit is DVID's ability to serve tiled image data (as opposed to 3D voxel blocks), as well as multi-resolution voxel blocks. Multi-resolution tile/block viewers such as Neuroglancer, BigDataViewer and CATMAID are all capable of using DVID's multi-resolution block streaming

APIs, with interactive speed. Unfortunately, current ilastik's viewer does not support multi-resolution data sources, and so does not exploit this feature, either. Multi-resolution support is, however, planned for the HBP web-based viewer, which we also plan to adapt to as a possible ilastik front-end.

In summary, DVID appears to be a decent choice as a data service if it is well supported by a viewer and multi-user versioned data editing is required.

6. Proposed Architecture and Roadmap

The proposed architecture is intended to be the subject of continual capability expansion over the course of the HBP. Each Phase is intended to deliver an incremental improvement in capabilities to satisfy a particular use case or set of use cases. To this end, it is essential that the architecture be both performant and flexible providing well defined, well supported API boundaries between services to allow replacement or substitution of services with different performance criteria or add-on functionality as both the implementation and use cases evolve with time.

Alignment of the viewer architecture strategy with image formats is based on the neuroimaging format standardisation document (MS5.4.6 preliminary version, updated version due M24). As it turns out, the most important 2D and 3D image standards that the viewer architecture will have to support are:

- NIfTI for volumetric data (T1, T2, activation and probability maps);
- GIfTI for mesh-based surface representations;
- TIFF/BigTIFF, JPEG, PNG or laboratory-wise flavoured HDF5 for microscopy data. *So that they can be handled efficiently in the object storage, it is important for large 3D HDF5 datasets to be stored in appropriately sized chunks that can be used during data ingest. This has to be communicated with the laboratories that produce large HDF5 datasets and documented in the image standardisation document (MS5.4.6).*

The compatibility between image formats and viewers will be provided through the HBP image service, which is envisioned to be based on DVID.There will also be a requirement for an ingest service that converts incoming supported file formats into the internal format of the image service. DVID provides such a system, but may require some adaptation to allow efficient handling of data in formats that it does not currently support natively.

Further, we will focus on accelerating large and mostly static datasets. These are especially large 2D tiles and strategic high-resolution volumes like the Big Brain. For such data, we rely on the Neuroglancer precomputed tile format for 3D data, and single-image DZI (DeepZoom) for 2D multiresolution data. The data can be rendered right away by Neuroglancer and OpenSeadragon based viewers.

The viewer infrastructure will support the following storage resources.

1. Federated object storage hosted by the HPAC platform.
   Current developments connect to CSCS's Pollux system as the first and representative system available. This system will be scaled at least to the needs of the next ~5 years of HBP planning.
2. GPFS-based storage in the HPC centres.
3. Client-side storage for small images.
   For some use cases, the system should allow users to display images from their local computers, most often overlaid with HBP datasets. This is especially required for neuroimaging, where users may want to overlay their own MRI scans with atlas data without bringing their data physically into the HBP ecosystem.

6.1 Main categories of viewers required

As evident from the use cases and the resulting requirements analysis, viewers will have to cover requirements of 1D, 2D or 3D neuroscience data. For each of the dimensions, the time domain can be added, resulting in 6 categories of data, of which 1D snapshot data does not require a viewer. In general, 1D data could easily be analysed, transformed and displayed using scientist friendly tools in the Collaboratory's Jupyter notebooks. This allows for much richer interactions than would be possible with a GUI viewer.

The viewer strategy and architecture will thus have to cover the remaining **5 categories** (green):

Table 3: Viewer strategy and architecture categories

Data type	Snapshot viewer needed?	Time Series viewer needed?	Server-side rendering needed (SGA1)
1D data	no	possibly	no
2D data	yes	yes	no
3D data	yes	yes	no

The viewer infrastructure includes the following viewer components.

- A multi-resolution 2D viewer for possibly large microscopic 2D imagery, with additional support for moderately sized vector graphics to display 2D annotations. The viewer will be based on *OpenSeadragon. The core development of this viewer is in T5.4.2.*
- A multi-resolution 3D viewer for possibly large volumetric images, with additional support for moderately sized meshes to display 3D segmentations like brain regions, surfaces, or individual microstructures. *This viewer will be based on Nehuba, HBP's extension of Neuroglancer. The core development of this viewer is in T5.4.3.*

This set of viewers will cover very frequent needs of currently existing custom solutions to navigate and interact with standard imagery. However, it will not attempt to replicate all expert functionalities that individual laboratories or researchers used for working with their data. For very specific individual workflows, we assume researchers continue using their own tools. Table 4 provides a detailed list of current solutions with regard to our considerations about whether to replace these tools by our core online components. For example, activity data will usually be previewed by activation maps, for example, and users will look at the time series from a Python notebook or similar expert tool that can connect to the NIP through an API.

Table 4: Currently used image viewers

Solution	Description	Used by or for	How does it work with NIP viewer infrastructure? Will it be replaced in the unified future setup or co-exist?
HBP developments			
HiBoP	Stand alone tool for iEEG data based on C++/Unity, developed by the team of J.-P. Lachaux	Visualising intracranial EEG data on individual 3D MRI	HiBoP will co-exist as an expert viewer for client-side installation. HBP is considering whether to implement interfaces so HiBoP can directly connect to HPB's Image service API. Some features will be replicated in Nehuba to allow previews of iEEG data that are visually consistent with HiBoP
VisNEST	System for visualising NeST simulations, using coupled multiple views and inter-process communication	Used to visualise dynamic parameters during a neural network simulation, such as calcium concentrations	The system is used for *in situ* visualisation of simulations running on clusters, a setup largely different from the NIP ecosystem. It will coexist with the NIP viewer infrastructure
Elephant visualisation component	Visualisation of elphys recordings for Elephant	Visualisation of activity data (spikes, LFP,...) and resulting analysis results from simulation and experiment (planned SGA2 component)	The Elephant visualisation component is planned as a stand-alone library to generate standardised graphics of datasets that are able to be represented in the Neo data model and for analysis results resulting from functionality of the Elephant library. It will co-exist with NIP viewers as a tool for experts performing data analysis. In addition, it is planned to investigate the feasibility of using this component in conjunction with other NIP viewer components to allow users to visualise activity data sets online, e.g., on a data landing page for activity data
ESPINA	Desktop software for interactive 3D segmentation of microstructure in electron microscopy	Used by neuroscientists to interactively extract e.g. neuron morphologies from EM data	This is a desktop application for image segmentation, which includes a viewer component. The EM data and segmented compartments typically used here can be visualised in Neuroglancer/Nehuba. Some of the segmentation functionality can be covered directly online through the

			NIP by ilastik, once it is tightly integrated
Nehuba	HBP atlas extensions to Neuroglancer, a web based volumetric image viewer	Visualisation of possibly large volumetric data and parcellations online. Already used for reference atlases and high-resolution templates like the Big Brain	Nehuba is a core part of the SP5 atlas tool suite, and currently being extended by interactive components to cover more use cases than browsing of reference atlases. This is the viewer that will include the components for interactive analysis with ilastik and be used for overlaying volumetric data with high-resolution atlases on the web
PLIviewer	Viewer developed at RWTH specifically for PLI data	Specific solution for visualising PLI connectivity data, currently used by PLI experts	HBP would like to include similar functionality in the Nehuba web-based viewer (see previous entry). A proposal to implement web-based visualisation of high-resolution connectivity was not accepted for the SGA2 workplan, but will be raised again for SGA3
BrainScales activity and network viewers	Specific viewer for monitoring neuromorphic computing	UHEI, BrainScales users	This is not an image viewer, and will co-exist as software for the neuromorphic computing community
SpiNNaker activity and network viewer	Specific viewer for monitoring neuromorphic computing	UMAN, SpiNNaker users	This is not an image viewer, and will co-exist as a software for the neuromorphic computing community
Community tools			
Amira	Commercial general-purpose biomedical 3D visualisation	Visualise 3D volumes and segmentations offline (LENS)	Amira will co-exist as an offline tool for individual experts. It can be applied to data downloaded from the NIP
Vaa3d	Open-source biomedical 3D visualisation	Visualise 3D volumes and segmentations offline (LENS)	Vaa3d will co-exist as an offline tool for individual experts. It can be applied to data downloaded from the NIP. The software is extensible with plugins, and an interface to connect to HBP services could be implement, but is not currently planned

Fiji	Open-source biomedical 3D visualisation and image processing toolbox	General stand-alone image viewing and processing tool set for the biomedical imaging community	Fiji will co-exist as an offline tool for individual experts. It can be applied to data downloaded from the NIP. The software is extensible with plugins, and an interface to connect to HBP services is feasible, but is not currently planned
MRIcroN, FSLView	Offline MRI viewer	Many neuroimaging researchers	As an offline solution only for small data volumes, MRIcroN, FSLView will co-exist. We will evaluate navigation in Nehuba against the user interaction used in these viewers, to facilitate as much as possible the use of atlas viewers for neuroimaging users
MayaVi	Python library and client for general purpose, highly customisable 3D visualisation	Some expert developers, data analysts in SP1, SP2	MayaVi is more of a developer's tool. The NIP viewer architecture is not intended to replace such highly customisable general purpose libraries, so they will coexist
Brainvisa Anatomist	multifaceted viewer developed at Neurospin	Neuroimaging researchers	Brainvisa is a complex, stand-alone neuroimaging tool suite. The included viewer will co-exist with HBP's online services
Neurolucida	Commercial stand-alone tool for neuron tracing, reconstruction, analysis, and 3D brain mapping	Some microscopy labs	This is a comprehensive tool suite specifically targeted to extract 3D neuronal structures from high-resolution 3D imagery. Some of its functionality will also be provided by the NIP ecosystem in upcoming SGAs, especially interactive segmentation for 3D brain mapping. As such, HBP's NIP will provide free and lower threshold access to these critical tools for the neuroscientific community
Anywave	Visualise MEG and EEG recordings	Used by some researchers in SP4	Anywave is a C++ based tool to allow the interactive browsing of EEG and MEG data time series. In the context of visualising electrophysiological data, it complements the planned Elephant Visualisation Component, which is a low-level library to visualise single unit and field potential data from microelectrode recordings and to provide views of analysis results performed on such data.

			The modular architecture of Anywave might allow a deeper integration of the tools via signal processing plugins
McGill BrainBrowser	Visualise MRI volumes online	SP8	Solution can in the long term be replaced by Nehuba, which can provide most of the features as well. Feature comparison to be done together with SP8
OpenSeadragon	Open source JS library for browsing multi-resolution 3D image data	SP1, SP5, SP2 for browsing microscopic slides	OpenSeadragon will be used as a component for 2D web-based atlas visualisation, complementing Nehuba/Neuroglancer as a more efficient solution for 2D high-resolution images

6.2 Phase 1

Phase 1 is intended to focus on the tools necessary for widespread dissemination via so-called released atlases. These include the Waxholm, Allen, Jubrain and BigBrain reference atlases along with any other atlases that are expected to have a heavy user base. See Figure 4.

Objectives

1) Reference atlas file specification – define the files needed to make an atlas and add the requisite support for those files to tools used in the phase 1 architecture.
2) Development of initial scripts to package a reference atlas for internet viewing.
3) Minimise unauthenticated access to atlases. Public and Read-only.
4) Reference atlas link(s) from the front-page of the new NIP page.

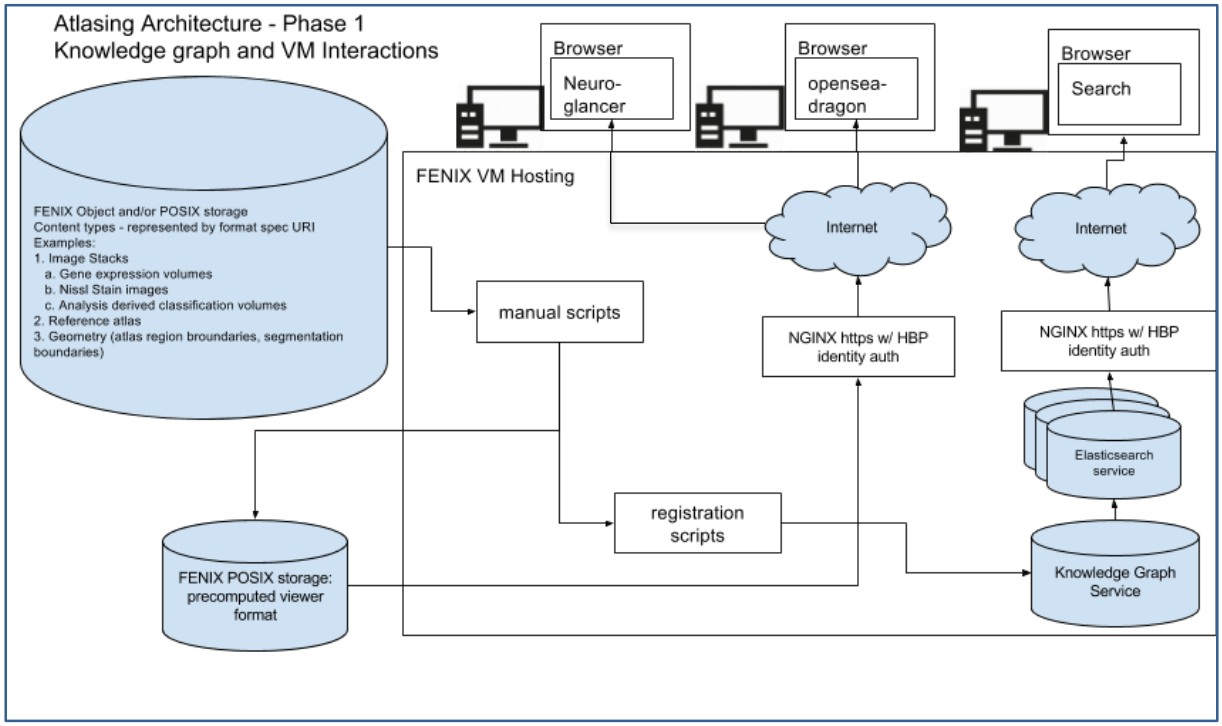

Figure 4: Strategy and architecture for data viewers - Phase 1

6.3 Phase 2

In this phase the goal is to increase the accessibility of atlases that are in-development using Navigator, the volume alignment tools or extracted features from ilastik workflows (whether initiated via ilastik-web or on a local cluster).

Make unreleased/in-progress atlas developments visible via the viewers.

1) Integration path for Oslo toolkit and ilastik.
2) Authenticated access to atlases. Read-only.
3) Authorisation controlled by the Collab, which owns the atlas.

This requires the following activities, broken down on a service by service level.

Navigator3

1) Support for OIDC authentication in navigator.
2) Port and testing of Navigator on PostgreSQL.
3) A deployment of Navigator to FENIX VMs using Postgres DB.
4) Add support for "release" of a Navigator project to the precomputed volume service.

DVID

1) Create a DVID app and bind a given DVID project to a Collab.
2) Enable OIDC authentication and Collab ACL authorisation.
3) Testing of DVID in production deployment environment.
4) Add support for "release" of a Navigator project to the precomputed volume service.

Ilastik

1) Add support for OIDC authenticated DVID to ilastik.
2) Test last two releases of ilastik with deployed DVID.

Neuroglancer, 3D

1) Load list of atlases or other datasets from Knowledge Graph.
2) Load an atlas' respective annotation hierarchy according to HBP atlas standard.
3) Display annotation hierarchy viewer in a side panel for reference atlases.
4) Enable annotation hierarchy viewer control functions (potentially: transparency control, visibility, hide others).
5) Improve 3d cut-plain view to be more intuitive to neuroscience and medical customers.
6) Authentication and authorisation support for OIDC.

OpenSeadragon, 2D

1) Load list of atlases or other datasets from Knowledge Graph.
2) Load an atlas's respective annotation hierarchy according to HBP atlas standard.
3) Display annotation hierarchy viewer in a side panel for reference atlases.
4) Enable annotation hierarchy viewer control functions (potentially: transparency control, visibility, hide others).
5) Authentication and authorisation support for OIDC.

Knowledge Graph

1) Provide a list of atlases via a permanent query that can be hardcoded into the HBP deployments of Neuroglancer.

The resulting architecture for Phase 2 will integrate the atlas building tools of Navigator and the analysis tools of ilastik with 2d and 3d viewers.

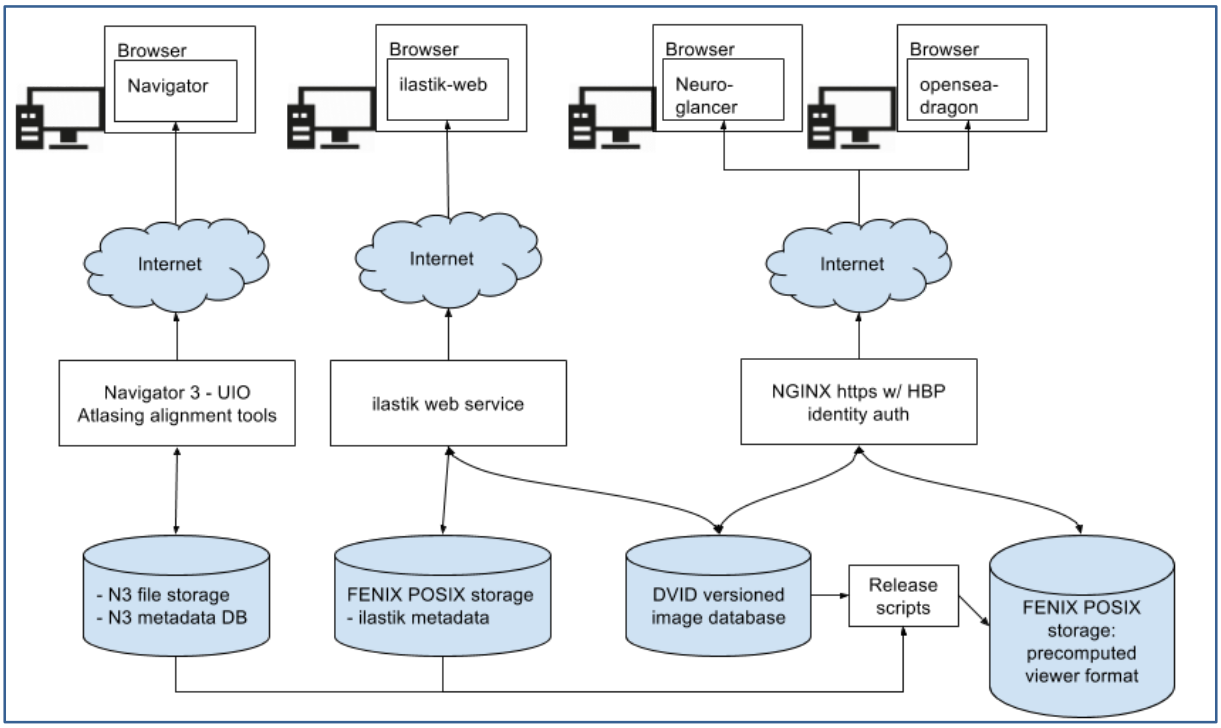

Figure 5: Strategy and architecture for data viewers - Phase 2.

6.4 Possible Future Phases

6.4.1 Multi-viewer, multi-controller coordination

SP5 expects to deploy a number of viewers, pragmatically selected for their applicability to specific use cases. A viewer for a slice might want to exchange or sync view information. A viewer for a movie may want to time sync with a view of MRI data. Additionally, it might be desirable to create novel control widgets outside of the viewer itself. Some of the proof of concepts implemented with another HBP project (https://github.com/HBPVIS/ZeroEQ) suggest how this might work and why it would be useful.

6.4.1.1 Integration strategy

In principle, the main impediments are security considerations of running such a system in a large-scale untrusted environment. Should the multi-viewer, multi-controller use case come to the forefront in a future phase, it would be necessary merely to extend ZeroEQ with appropriate encryption, authentication, and authorisation support. Subsequently, applications with a clear need for multi-viewer or multi-controller support could be loosely coupled using ZeroEQ or equivalent. This would open a wide range of extension scenarios.

6.4.2 Server-side rendering and High-fidelity viewers

As outlined above, the HBP does not have use cases that demonstrate a clear need for server-side rendering in the current phase.

In the future there are a number of places which could change this picture. If the demand for the interactive analysis of the large-scale detailed neuron simulations grows, high-resolution remote visualisation may become an important tool for understanding network simulations.

Similarly, if the need for extensive web visualisation of faster-than-real-time neural simulation grows, server-side rendering tools might be necessary to visualise the massive quantities of data those simulations are capable of producing.

In the atlasing domain, it is expected that HBP might benefit from the development of server-side rendering. One strength of HBP's atlases is the rich coverage of connectivity at multiple scales, ranging from single axon measurements in 2-photon-imaging through polarised light imaging at the micro- and mesoscales up to highfield and clinical DTI. In order to allow visual exploration of such multiscale setups on the web, we like to include visualisation of fibre tracts, orientation vectors, and advanced glyphs for tensors. While web viewer solutions for such kind of data are now available (e.g. http://www.nmr.mgh.harvard.edu/~rudolph/webgl/brain_viewer/brain_viewer.html), this use case would require a multi-resolution streaming strategy for dense fields of surface-based objects. Such a visualisation engine is in principle feasible, but would need to be developed. An alternative solution for multi-scale connectivity, with a possible less optimal user experience, is server-side remote rendering as described above.

6.4.3 Big-data analysis frameworks (Spark, etc.)

Interestingly enough, the data science community has focused their attention for large-scale analysis problems on frameworks like Spark. The result is essentially a Jupyter notebook with massive aggregate memory and IO capacity and the visualisations of choice are largely 2D. While 2D graphs may not look as pretty, the industry momentum in data science suggests that appropriate projections of data to 2D visualisations is a *more* valuable tool for data scientists. Such visualisations are trivially web-ready.

Moving in the same direction, the need for ever higher performance from popular tools in the Collaboratory is already starting to become clear. As researchers push the envelope with the science they share in the Collaboratory's Jupyter notebooks, there is an obvious demand for an interactive computing environment which surpasses the limits of a single machine.

The most obvious contender for this crown at the moment is a Jupyter notebook running in front of a large multi-tenant Spark installation. Spark is seeing widespread application across

scientific domains from large-scale electrophysiology analysis (https://www.youtube.com/watch?v=Gg_5fWIlfgA) to bone microstructure analysis (https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-015-1617-y).

It is clear the precomputed volume format for the reference atlases is horizontally scalable using standard HTTP CDN approaches, but the precomputed volume format will not be used for large-scale read-write workflows. Here the architecture above promises to deliver. Janelia Farms is the driver of the Fly-EM project (and by extension the DVID store) and has made extensive use of DVID as the primary datastore for large-scale Ilastik and Spark based image processing workflows. See https://github.com/janelia-flyem/DVIDSparkServices for more details on the current state of this work.

7. Conclusions

The image service has to provide streaming of 2D and 3D image data to the viewers for use cases where conversion into precomputed formats is inefficient or not appropriate. This is especially the case for *ad-hoc* visualisation, e.g. for monitoring interactive image analysis online, as in ilastik, or user data coming in. We will fill this gap based on the DVID image service, and evaluate its performance in SGA2. The worst outcome of this evaluation is that DVID is not efficient enough to deliver 3D data in multi-resolution to our viewers. In this case, we will extend DVID in this regard in SGA2/3, or implement a replacement component. We expect to be able to dedicate more resources to the integration of the image service in SGA2 compared to SGA1.