BIDS Extension Proposal XXX (BEPXXX):

The BIDS Computational Model Specification

version 0.9.0 (working copy)

Available under the CC-BY 4.0 International license.

Michael Schirner <[michael.schirner@charite.de](mailto:michael.schirner@charite.de)> & Petra Ritter <[petra.ritter@charite.de](mailto:petra.ritter@charite.de)>

|  |
| --- |
| This document contains a draft of the Brain Imaging Data Structure standard extension. It is a community effort to define standards in data / metadata. This is a working document in draft stage and any comments are welcome.  This specification is an extension of BIDS, and general principles are shared. The specification should work for many different settings and facilitate the integration with other imaging methods.  To see the original BIDS specification, see [this link](https://docs.google.com/document/d/1HFUkAEE-pB-angVcYe6pf_-fVf4sCpOHKesUvfb8Grc/edit). This document inherits all components of the original specification (e.g. how to store imaging data, events, stimuli and behavioral data), and should be seen as an extension of it, not a replacement. |

# 1. Overview

We propose a data structure schema for biological neural network computer models that aims to be generically applicable to all kinds of neural network simulation software, mathematical models, computational models, and data models. Importantly, we not only propose suggestions for a BIDS schema for computer models, but we also propose extensions to the entire BIDS standard that solve several other problems.

We start out by addressing the problem **filename lengths and overspecialization.** The idea to put the characteristics of a dataset into the filename to disambiguate the files is appealing and works great for simple data sets and flat ontologies with a small number of interfaced software products, classes of models and concepts. However, in practice, the maximum filename length is quite limited in Linux, Windows, OS X (e.g. 256 characters in Windows for the full path), which is not much considering the multitude of parameters that can be involved in computational models. There are so many characteristics involved in these data sets that the maximally allowed length is easily reached and the defining characteristic is buried somewhere in a swarm of key-value pairs, which makes the filenames hard to read. Without rigorous visual parsing of every key-value pair one easily misses the defining characteristic of a particular file. In computer modelling practice, often many different subtle variants of a simulation are quickly produced, e.g. to test the effect of different parameter settings, which, given the many parameters of these models, would lead to enormously long filenames if the distinction between data sets is based on plain filename key-value schemas without any encoding.

This problem is closely related to the problem of **overspecialization**: BIDS formats seem to be usually developed with one or a few specific software products or concepts in mind as use cases. This leads to situations where the key-value pairs of file names are tuned to a narrow class of software or concepts that cannot be used outside of that specific framework. This is in contrast to the idea of BIDS of having a generic interface that is independent of a specific product, data format or narrow concept. When defining BIDS extensions, instead of interpreting the data again from the specific conceptual vantage point of a specific product or class of concepts, we could also ask what generic patterns and common structures underlie the specific data formats. A closer look reveals that the underlying data types of neural models are often quite simple and conceptually often very similar or even identical.

Therefore, in order to find a **solution**, we suggest converging towards the simple and basic data types and structures that underlie these neural computer models, which are: ***network graphs****,* ***systems of mathematical equations****,* ***computer code****, 1d vectors and 2d data matrices*[[1]](#footnote-0). Going one level of abstraction lower we may divide the two latter objects again into more spatially-oriented and more temporally-oriented objects like ***time series***, ***maps*** and ***coordinates***(or coordinate systems). To store all of these, instead of requiring complex formats, all we need are basic **tsv files**, **JSON files and XML files for model equations**.

Next, in order to disambiguate similar files we do not rely on lists of identifying characteristics -- we either end up with long filenames or we fail to identify a subset of characteristics that is generically applicable. We propose to go the opposite way: **short and concise filenames that communicate the contents at a high-level with a simple and intuitive label, together with a unique ID to distinguish files, and an extensive metadata specification in the sidecar JSON**. That is, we suggest that all the metadata is stored in sidecar JSON files and only a single characteristic, catchy and concise freeform \_desc-<label> is used as a human-readable disambiguating identifier. Furthermore, **one additional key-value pair is added to all BIDS files**, which we called \_id-<id> in the following, and which is about five to seven alphanumeric characters long. Here comes the kicker: this id not only serves to distinguish files, but it serves as a tool to increase confidence about data integrity, authenticity and validity, because: **the id of each data-file is a hash of its sidecar JSON metadata file**. This hash acts like a **checksum** of the metadata file, which means, with a few simple hashes we can **instantly check whether the information in each metadata file is correct and whether the entire data-metadata structure of the data set is valid**. This hash can be based on the entire metadata information, or a subset that is related to the provenance of the data and that is assumed to stay fixed. The upcoming BIDS checkers of the future would therefore be enabled to automatically check the validity and integrity of every data set with one simple operation. Also, the BIDS checker would serve to make the data set consistent in the first place without requiring the researcher to manually compute all the hashes and also to be backwards compatible with the current and previous BIDS standards. On its first run over a newly-created data set, it will tag all filenames with the appropriate hash id and therefore harmonize and integrate the data set. When people set up a data set they do not need to worry about the ids -- they just give them either no or an arbitrary freeform ID and the BIDS checker automatically creates a valid data set with proper ids. From that point on, any tampering with this data set or the introduction of any inconsistency or error will be instantly recognized!

Note that one can easily extend this consistency checking over the entire data, by adding checksums of the main data files to their sidecars, which makes it possible to instantly check that a data set is what it is expected to be or whether it was changed in any way.

Now we change gears and go one step further: tagging every data file with an id has advantages that would open up solutions for **worldwide provenance tracking and unique identifiability of each BIDS data set**. Now that we have verifiably consistent data-metadata structures and (unique) identifiers, we can exactly track the worldwide adventures of every single data set by adding one universally unique identifier (UUID) to each sidecar file.[[2]](#footnote-1) Either the filename ID or a unique ID inside the sidecar could be made such that all BIDS files worldwide become uniquely identifiable -- by using a hash function that hashes the metadata into some small unique key, so it has the additional advantage that we can always verify the authenticity and correctness and self-consistency of the metadata. The integrity of the entire directory schema and data/metadata organization and association can be automatically and instantly verified in this manner. The initial BIDS checker run would not only make a data set self-consistent, but it would also register that data set in a global database which from that point on explicitly tracks the whereabouts and transformations that happened to this data set. Each time a data set is transformed, the BIDS checker updates the data set with proper ids and also its index card in the registry. This addresses many problems that are currently hindering research with regard to worldwide data sharing, provenance and accountability tracking and may even lay the foundation for solutions that integrate data sets with their legal basis for sharing, sharing agreements, clarifies ownerships and other roles with regard to data protection laws and solutions for secure and lawful sharing of data worldwide. This could culminate in a global authoritative database and indexing system of neuroscience results with provable authenticity, yielding a world-wide record of annotated and verified data sets that are internally consistent and that can be used to exactly track the evolution of data sets and their metadata. Combined with security mechanisms like encryption and authentication, the BIDS standard may at some point reincarnate as the all-in-one solution for data standardization, provenance tracking, data security and lawful data exchange that respects the privacy of data subjects.

This is, of course, “a lot” and the above is merely an outlook of what may become possible. In the following, we focus on some specific schema principles, many of which can be used, whether the entire concept is adopted, or not.

# 2. Motivating principles

We primarily address the problem of reproducibility: it is often not possible to reproduce the results of studies, because important details necessary for replication are not explicitly described (e.g. parameter values, procedures) or difficult to determine (e.g. poorly documented custom code). When computers are involved, achieving full reproducibility would require the exact code and system on which the model was computed. Since long-term maintenance and distribution of the system on which the study was produced is infeasible, we here focus on rigorously specifying the involved software as well as the underlying mathematical and biophysical theory. **This requires an explicit formal specification of the mathematical equations and physical concepts, but also of the particular software implementations used for producing the result, together with function definitions, algorithms, parameter and variable settings.** In order to guarantee the long-term existence of software it must be versioned in a way that makes it uniquely identifiable and archived in long-term repositories. Since software often gets quite large it is impractical to store copies of the programs along with the results and rather the long-term repositories must be referenced from the metadata. **Furthermore, parameter values used for a specific simulation result must be stored in JSON metadata files that accompany the main simulation result data files.** For specifying equations we use the domain-specific language RateML[[3]](#footnote-2) which is an extension of LEMS[[4]](#footnote-3) for which there exist automatic code generators that transform the XML syntax into high-performance machine code for CPU or CUDA code for GPU.

An important core principle of this Extension Proposal is that it should be generally applicable to all kinds of neural simulators. Its design is such that it specifies common principles among neural network model simulations generically and pragmatically. This involves not defining hundreds of filetype classes for special use cases (“Who will use this atlas still in five years from now? Is it really necessary to have a data standard for this specific way of creating a surface/parcellation/atlas/etc. Or does it seem like it will be replaced anyway in the near future?”). Instead, data structures are generic and boiled down to the essential data types that neural network models need: model equations, network graph descriptions and parameter values. In addition, generic containers for time series and maps (tabular data) are proposed.

Importantly, simulation parameters are stored in each individual JSON file that accompanies a simulation result. Although this may be redundant, it makes the data set self-consistent, modular and less error-prone. Another consideration is that interface and parser designers should not have to search through extensive definitions of filenames and metadata schemas or have to constantly re-design, update and implement hundreds of very similar and only marginally different schemas.

# 3. Directory structure

The most important input for a neural network model is a description of the network edges (usually connection weights and connection distances), which happens usually in the form of connectivity matrices or sparse representations thereof. Next comes a description of the nodes connected by the network edges. These nodes can be single neuron models or models of neural populations. The last ingredient for a full computational model specification are the parameters, initial state and possible inputs. The primary outputs of these simulations are time series of node activity that is then often transformed into maps, like functional connectivity matrices. Data and metadata files are associated via their ids. It is allowed that multiple data files belong to the same data structure and are annotated by the same metadata file. For example, a surface would be specified by “coordinate” (coord/) files of different subtypes (vertices, faces, normals) and associated with their corresponding metadata file via a common <id>, and optionally, a common desc-<label>. For example:

sub-pat4\_desc-FSpipev2agg4\_coord-vertices\_id-32fa3t4a.tsv

sub-pat4\_desc-FSpipev2agg4\_coord-faces\_id-32fa3t4a.tsv

sub-pat4\_desc-FSpipev2agg4\_coord-normals\_id-32fa3t4a.tsv

sub-pat4\_desc-FSpipev2agg4\_coord\_id-32fa3t4a.json

Networks are stored in the subfolder net/ and identified by the key net- with possible subtypes distances and weights. A JSON sidecar file specifies details like the used atlas or preprocessing pipeline. Mathematical equations are specified in RateML format (XML-based), stored in the eq/ directory and identified with the keyword eq . The label of eq should be a short string to distinguish the type of equations: nm stands for “neural model” (alternatives could, e.g., ml for machine learning model, cg for computational graph, cm for cognitive model, etc.). Equation files have a special role among these file formats, because they are the only file type that uses an XML format instead of JSON, because it is based on LEMS[[5]](#footnote-4),[[6]](#footnote-5),[[7]](#footnote-6) for which there exist automatic machine code generators for CPU[[8]](#footnote-7) and GPU[[9]](#footnote-8). The desc keyword follows the BIDS-Derivatives convention, and allows a freeform description (but should be kept succinct) to distinguish between the important characteristics of these files like different ways of preprocessing or different spaces. net stands for “network”. Coordinate files (keyword and folder coord) can contain any kind of coordinates, e.g. sensors, surface vertices, region centres, etc. Time series (folder name and keyword ts) can contain any kind of time-varying activity, e.g. EEG, LFP, stimulation, artificial time series, etc. Parameters of simulation results are stored in the json sidecar of the simulation result.

Listing 1 shows a full example of the proposed computational model directory structure.

sub-<label>/

net/

sub-<label>[\_desc-<label>]\_net-distances\_id-<id>.tsv

sub-<label>[\_desc-<label>]\_net-weights\_id-<id>.tsv

sub-<label>[\_desc-<label>]\_net\_id-<id>.json

*sub-pat01\_desc-MMPtestv3\_net-weights\_id-c479d52f.tsv*

*sub-pat01\_desc-MMPtestv3\_net-distances\_id-c479d52f.tsv*

*sub-pat01\_desc-MMPtestv3\_net\_id-c479d52f.json*

*sub-pat01\_net-weights\_id-test3.tsv*

*sub-pat01\_net\_id-test3.json*

coord/

sub-<label>[\_desc-<label>]\_coord-vertices\_id-<id>.tsv

sub-<label>[\_desc-<label>]\_coord-faces\_id-<id>.tsv

sub-<label>[\_desc-<label>]\_coord-normals\_id-<id>.tsv

sub-<label>[\_desc-<label>]\_coord-labels\_id-<id>.tsv

sub-<label>[\_desc-<label>]\_coord\_id-<id>.json

sub-<label>[\_desc-<label>]\_coord-<label>\_id-<id>.tsv

sub-<label>[\_desc-<label>]\_coord-<label>\_id-<id>.json

*sub-hcp12\_desc-hcpstudy4run3\_coord-meg\_id-4384e12a.tsv*

*sub-hcp12\_desc-hcpstudy4run3\_coord-meg\_id-4384e12a.json*

*sub-hcp12\_desc-hcpstudy4run3\_coord-stim\_id-run3stim.tsv*

*sub-hcp12\_desc-hcpstudy4run3\_coord-stim\_id-run3stim.json*

ts/

sub-<label>[\_desc-<label>]\_ts-<label>\_id-<id>.tsv

sub-<label>[\_desc-<label>]\_ts-<label>\_id-<id>.json

*sub-con3\_desc-stimat8sv4\_ts-sim\_id-3dsa45.tsv*

*sub-con3\_desc-stimat8sv4\_ts-sim\_id-3dsa45.json*

map/

sub-<label>[\_desc-<label>]\_map-<label>\_id-<id>.tsv

sub-<label>[\_desc-<label>]\_map-<label>\_id-<id>.json

*sub-con3\_map-lfm-eeg\_id-test1.tsv*

*sub-con3\_map-lfm-eeg\_id-test1.json*

*sub-con01\_map-FC-sim\_id-339dda.tsv*

*sub-con01\_map-FC-sim\_id-339dda.json*

eq/

eq-<label>[\_desc-<description>]\_id-<id>.xml

*eq-nm\_desc-RWW\_id-fb8c577a.xml*

code/

code-<label>[\_desc-<description>]\_id-<id>.<lang\_ext>

code-<label>[\_desc-<description>]\_id-<id>.json

*code-nm\_desc-RWW\_id-fa3ac7da.c*

*code-nm\_desc-RWW\_id-fa3ac7da.json*

README

participants.tsv

dataset\_description.json

CHANGES

Listing 1: Directory structure (examples in gray italics).

# 4. File names and file contents

The following table (Table 1) provides some examples how typical entities from computational brain modelling could be represented under the proposed standard.

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity** | **Filename schema** | **Format** | **Description** |
| Network edges | sub-<label>[\_desc-<label>]\_net-distances\_id-<id>.tsv | n x n | distances |
| sub-<label>[\_desc-<label>]\_net-weights\_id-<id>.tsv | n x n | weights |
| sub-<label>[\_desc-<label>]\_net\_id-<id>.json | json | metadata |
| Surface topology | sub-<label>[\_desc-<label>]\_coord-vertices\_id-<id>.tsv | n x 3 | Cartesian coordinates |
| sub-<label>[\_desc-<label>]\_coord-faces\_id-<id>.tsv | n x k | faces of topological manifolds (e.g. triangles) |
| sub-<label>[\_desc-<label>]\_coord-normals\_id-<id>.tsv | n x 3 | normal vectors |
| sub-<label>[\_desc-<label>]\_coord-labels\_id-<id>.tsv |  | vector of text string labels |
| sub-<label>[\_desc-<label>]\_coord\_id-<id>.json | json | metadata |
| Sensor or stimulation locations | sub-<label>[\_desc-<label>]\_coord-<sensor>\_id-<id>.tsv | n x 3 | Cartesian coordinates |
| sub-<label>[\_desc-<label>]\_coord-<sensor>\_id-<id>.json | json | metadata |
| Time series | sub-<label>\_ses-<label>[\_desc-<label>]\_ts-<label>\_id-<id>.tsv | t x (n+1) | Temporal evolution of *n* objects over *t* time points. The first column must contain the time points. |
| sub-<label>\_ses-<label>[\_desc-<label>]\_ts-<label>\_id-<id>.json | json | metadata |
| Maps | sub-<label>[\_desc-<label>]\_map-<label>\_id-<id>.tsv | n x m | Maps can contain numbers \*and\* strings. For example lists of region or sensor names or operators like lead-field matrices. |
| sub-<label>[\_desc-<label>]\_map-<label>\_id-<id>.json | json | metadata |
| Model equations | eq-nm[\_desc-<description>]\_id-<id>.xml | XML | Model equations. These are referenced from simulation result sidecar files (where the used parameter values are specified) and can be automatically converted to machine codes. |
| Code | code-nm[\_desc-<description>]\_id-<id>.<ext> | Language-specific | Code used to produce neural model simulations. |
| code-nm[\_desc-<description>]\_id-<id>.json | json | metadata |

Table 1. Exemplary model entities and their representation.

# 5. List of supported filename key-value pairs

In the following we provide an initial list of supported key-values pairs.

|  |  |  |
| --- | --- | --- |
| **Keyword** | **Supported values** | **Description** |
| id | hashed JSON sidecar (alternatively: free-form) | This ID distinguishes files of the same type and, if the hash version is used, can be used to verify the integrity of the data set. An additional UUID in the metadata makes the file uniquely identifiable. |
| net | distances | Length or time delay associated with an edge |
| weights | Weight of an edge |
| coord | vertices | Vertices of a discretized manifold (e.g. a surface triangulation) |
| faces | Faces of manifold |
| normals | Normal vectors of manifold vertices |
| meg | Locations of MEG squids |
| eeg | Locations of EEG electrodes |
| stim | Location of stimulation electrodes |
| ts | sim | Simulated time series |
| emp | Empirical time series |
| stim | Stimulation time series |
| art | Artificial time series |
| map | lfm-eeg | EEG lead-field matrix |
| lfm-meg | MEG lead-field matrix |
| area | Surface areas |
| volume | Body volumes |
| fc-sim | Simulated FC |
| fc-emp | Empirical FC |

Table 2. List of supported values for filename schema keys.

# 6. JSON sidecar metadata files

We suggest to use as much of the existing schemas and vocabulary regarding entities, relationships between entities and actions from schema.org

In the following we provide example listings for specific JSON metadata file types, but first we give an overview over keywords defined herein and their requirement levels.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | net | coord | ts | map | code |
| ID |  |  |  |  |  |
| UUID |  |  |  |  |  |
| SubjectID |  |  |  |  |  |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Description** |  | | | | | | | Name |  |  |  |  |  |  | | Type |  |  |  |  |  |  | | SubTypes |  |  |  |  |  |  | | TypeLong |  |  |  |  |  |  | | Version |  |  |  |  |  |  | | | | | | |
| Desc |  |  |  |  |  |
| DataFiles |  |  |  |  |  |
| Units |  |  |  |  |  |
| VolumeUnit |  |  |  |  |  |
| SurfaceUnit |  |  |  |  |  |
| LengthUnit |  |  |  |  |  |
| Atlas |  |  |  |  |  |
| Surface |  |  |  |  |  |
| Space |  |  |  |  |  |
| TimeSteps |  |  |  |  |  |
| Format |  |  |  |  |  |
| Notes |  |  |  |  |  |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **SourceData** |  | | | | | | | UUID |  |  |  |  |  |  | | ID |  |  |  |  |  |  | | DOI |  |  |  |  |  |  | | URL |  |  |  |  |  |  | | DataFiles |  |  |  |  |  |  | | | | | | |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Nodes** |  | | | | | | | Count |  |  |  |  |  |  | | ID |  |  |  |  |  |  | | FullName |  |  |  |  |  |  | | IsCortical |  |  |  |  |  |  | | Volume |  |  |  |  |  |  | | Center\_X |  |  |  |  |  |  | | Center\_Y |  |  |  |  |  |  | | Center\_Z |  |  |  |  |  |  | | | | | | |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Model** |  | | | | | | | Equations |  |  |  |  |  |  | | Network |  |  |  |  |  |  | | Code |  |  |  |  |  |  | | IntegrationScheme |  |  |  |  |  |  | | Parameters |  |  |  |  |  |  | | Variables |  |  |  |  |  |  | | | | | | |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Publication** |  | | | | | | | DOI |  |  |  |  |  |  | | URL |  |  |  |  |  |  | | Type |  |  |  |  |  |  | | Title |  |  |  |  |  |  | | Authors |  |  |  |  |  |  | | Volume |  |  |  |  |  |  | | Number |  |  |  |  |  |  | | Pages |  |  |  |  |  |  | | Year |  |  |  |  |  |  | | Publisher |  |  |  |  |  |  | | | | | | |
| Parameters |  |  |  |  |  |
| Variables |  |  |  |  |  |
| Identities |  |  |  |  |  |
| EqFile |  |  |  |  |  |

Table 3: Requirement levels for different keys. (Legend: MUST | MUST IF APPLICABLE | RECOMMENDED | OPTIONAL | NOT APPLICABLE)

## 6.1. net/

Networks store the graph structure of neural network models.

|  |
| --- |
| sub-HCP100206\_desc-pipeline3atlas4\_net\_id-c479d52f.json |
| {  **"ID"**: "c479d52f",  **"UUID"**: "c165a630-2813-4d0b-a705-68c47549c302",  **"SubjectID"**: "HCP100206",  **"Desc"**: "pipeline3atlas4",  **"DataFiles"**: [  "sub-HCP100206\_desc-pipeline3atlas4\_net-weights\_id-c479d52f.tsv",  "sub-HCP100206\_desc-pipeline3atlas4\_net-distances\_id-c479d52f.tsv",  ],  **"Description"**: "Structural Connectivity",  **"VolumeUnit"**: "mm^3",  **"LengthUnit"**: "mm",  **"Atlas"**: "Glasser",  **"Notes"**: "Constructed by pipeline 3 with Atlas 4, details can be found at: ...",  **"Nodes"**: {  **"Count"**: 3,  **"ID"**: ["PFC\_L","OCC\_R","BSTEM"],  **"FullName"**: ["Left Prefrontal Cortex", "Right Occipital Cortex", "Brainstem"],  **"IsCortical"**: [**true**, **true**, **false**],  **"Volume"**: [100, 90, 50],  **"CenterX"**: [3.1, 4.2, 5.3],  **"CenterY"**: [6.2, 7.5, 8.2],  **"CenterZ"**: [9.1, 10.2, 11.1]  },  **"SourceData"**: {  **"UUID"**: "12036e75-618b-4ad6-844d-9f0da45553bf",  **"DOI"**: "doi:10.3972/water973.0145.db",  **"URL"**: "https://kg.ebrains.eu/.../c165a630-2813-4d0b-a705-68c47549c302"  }  } |

## 6.2. coord/

Coordinates are containers for all data that contains spatial, topological, temporal, etc. coordinates. For example, points and triangles and normals of a surface triangulation, EEG/MEG sensor coordinates, stimulation electrode locations, or points in time.

|  |
| --- |
| sub-pat1\_desc-FSnative\_coord\_id-k3923md2.json |
| {  **"ID"**: "k3923md2",  **"UUID"**: "e9ce0d81-a28a-4f4a-88f4-904539ffaa20",  **"SubjectID"**: "Pat1",  **"Desc"**: "FSnative",  **"DataFiles"**: [  "sub-pat1\_desc-FSnative\_coord-vertices\_id-k3923md2.tsv",  "sub-pat1\_desc-FSnative\_coord-faces\_id-k3923md2.tsv",  "sub-pat1\_desc-FSnative\_coord-normals\_id-k3923md2.tsv",  "sub-pat1\_desc-FSnative\_coord-labels\_id-k3923md2.tsv"  ],  **"Surface"**: "pial",  **"Space"**: "fsnative",  **"Description"**: {  **"Name"**: "Coordinates",  **"Type"**: "Surface"  },  **"SourceData"**: {  **"UUID"**: "12036e75-618b-4ad6-844d-9f0da45553bf",  **"DOI"**: "doi:10.3972/water973.0145.db",  **"URL"**: "https://kg.ebrains.eu/.../c165a630-2813-4d0b-a705-68c47549c302"  }  } |

## 6.3. ts/

This is a container to store all kinds of time series, empirical as well as simulated. If they are based on empirical data, source data sets should be specified. If they are based on model simulations the underlying model and parameters must be specified. Parameter values can be of type scalar or vector. If they are type vector, its length must be equal to the value of the field *nodes.count* in the Network JSON sidecar file. For all parameters and variables that are not specified in this file the default values, respectively initial values, as specified in the model file, are used. For the keyword “IntegrationScheme” a simple string value like “Heun” or “RK4” is likely not be enough for full reproducibility. Here, again a concrete mathematical and computationa implementation must be specified. So "IntegrationScheme" is itself a BIDS “eq/” and an associated BIDS “code/”.

|  |
| --- |
| sub-con3\_ses-testsim1\_desc-stimat8sv4\_ts-sim\_id-3dsa45.json |
| {  **"ID"**: "3dsa45",  **"UUID"**: "dafeaba8-51b8-4ffa-9e76-8b52148dbf17",  **"SubjectID"**: "con3",  **"Session"**: "testsim1",  **"Desc"**: "stimat8sv4",  **"DataFiles"**: [  "sub-con3\_ses-testsim1\_desc-stimat8sv4\_ts-sim\_id-3dsa45.tsv"  ],  **"Description"**: {  **"Name"**: "Time Series",  **"Type"**: "Simulated"  },  **"Notes"**: "There was a strange warning when ...",  **"Units"**: "s",  **"Timesteps"**: 100000,  **"Model"**: {  **"Equations"**: "eq-nm\_desc-RWW\_id-fb8c577a.json",  **"Network"**: "sub-HCP100206\_desc-pipeline3atlas4\_net\_id-c479d52f.json",  **"Code"**: "code-nm\_desc-RWW\_id-fa3ac7da.c",  **"IntegrationScheme"**: "HeunStochastic",  **"Parameters"**: {  **"W\_E"**: 1.0,  **"J\_NMDA"**: [1.0, 2.1, 3.2]  },  **"Variables"**: {  **"S\_i^E"**: 0.0  }  }  } |

## 6.4. map/

Maps are generic containers for all other kinds of one- or two-dimensional data like tables, arrays, matrices, vectors. For example, maps can be projection matrices or, in general, all kinds of matrix operators for transformations. Other examples include dictionaries or maps that associate different data elements, like a map that associates vertices of a cortical surface with the name of the brain region at that position. Also, simulation results that are not time series or coordinates can be stored here, like functional connectivity matrices or power spectral densities. Like in the case of ts/, if the data is based on empirical data source data sets should be specified. If it is based on simulated data, the model must be specified.

|  |
| --- |
| sub-con3\_map-lfm-eeg\_id-test1.json |
| {  **"ID"**: "test1",  **"UUID"**: "487180cd-c326-4c2c-9447-5440d7862622",  **"SubjectID"**: "con3",  **"DataFiles"**: [  "sub-con3\_map-lfm-eeg\_id-test1.tsv"  ],  **"Description"**: {  **"Name"**: "Map",  **"Type"**: "Lead-field matrix",  **"SubTypes"**: [ "EEG" ]  },  **"Notes"**: "Detailed description of preprocessing: sLORETA was used to...",  **"Units"**: "mV"  } |

|  |
| --- |
| sub-con01\_map-FC-sim\_id-339dda.json |
| {  **"ID"**: "test2",  **"UUID"**: "6bacf1f3-96da-40a6-8316-c7e0b7b0bfc8",  **"SubjectID"**: "con3",  **"DataFiles"**: [  "sub-con3\_desc-firstrun\_map-node2vertex\_id-test2.tsv"  ],  **"Description"**: {  "Name": "Map",  "Type": "vertex2node"  },  **"Notes"**: "The mapping was ...",  **"SourceData"**: [  {  **"ID"**: "3dsa45",  **"UUID"**: "dafeaba8-51b8-4ffa-9e76-8b52148dbf17",  **"DOI"**: "doi:10.3972/water973.0145.db",  **"URL"**: "<https://kg.ebrains.eu/.../c165a630-2813-4d0b-a705-68c47549c302>",  **"Notes"**: "This data set was ..."  }  {  **"ID"**: "a6a803fb",  **"UUID"**: "3d0755b7-8a45-4be5-b338-63cbfce3081f",  **"DOI"**: "DOI: 10.7554/eLife.55444",  **"URL"**: "file[://../../map/sub-con3\_map-atlas\_id-a6a803fb.json](https://kg.ebrains.eu/.../c165a630-2813-4d0b-a705-68c47549c302)",  **"Notes"**: "123"  }  ]  } |

## 6.5. code/

This JSON schema specifies metadata of Code files.

|  |
| --- |
| code-nm\_desc-RWW\_id-fa3ac7da.json |
| {  **"ID"**: "fa3ac7da",  **"UUID"**: "5c89d530-75ab-4afb-ba96-edc013f58f1e",  **"Desc"**: "RWW",  **"DataFiles"**: [  "code-nm\_desc-RWW\_id-fa3ac7da.c"  ],  **"Description"**: {  **"Name"**: "Code",  **"Type"**: "NM",  **"TypeLong"**: "Neural Model",  **"Version"**: 1.0  },  **"Notes"**: "This version contains a bug at line ...",  **"EqFile"**: "eq-nm\_desc-RWW\_id-fb8c577a.json"  } |

# 7. Model files: eq/

Equation files have a special role among the used file formats, because they are the only file type that uses an XML syntax instead of JSON, because it is based on LEMS[[10]](#footnote-9),[[11]](#footnote-10),[[12]](#footnote-11) for which there exist automatic code generators for CPU[[13]](#footnote-12) and GPU[[14]](#footnote-13). LEMS provides a compact, minimally redundant, human-readable, human-writable, declarative way of expressing models of biological systems. The LEMS format has been adopted and altered to match TVB model names. With the associated parser and transformer modules model XML files are converted to Python code for CPU and CUDA code for GPU.

The component type "derivatives" supports the specification of model equations in the form of time derivatives. The expression parser embedded in the LEMS library allows the following function expressions for the value attributes of the constructs: 'exp', 'log', 'sqrt', 'sin', 'cos', 'tan', 'sinh', 'cosh', 'tanh', 'abs' and 'ceil'. The following operands can be used to specify the mathematical expressions: '+', '-', '\*', '/', '^', and '~'. The power symbol in the expression fields needs to be entered within curly brackets, e.g.: {x^2}. Comparison operators must be specified with escape sequences, e.g. '\&lt(=);' for the "less than or equal to" and '\&gt(=);' for the "greater than or equal to" operator. See the original publications introducing LEMS[[15]](#footnote-14),[[16]](#footnote-15) and the GitHub repositories of the LEMS2TVB parser[[17]](#footnote-16),[[18]](#footnote-17) for more documentation. The following example shows the specification of the ReducedWongWang model[[19]](#footnote-18):

|  |
| --- |
| eq-nm\_desc-Deco2014\_id-fb8c577a.xml |
| <**Lems**  description="A number of ComponentTypes for rate based/population models.  Base type of any cell/population which has a (dimensionless) rate \_R.  Reduced Wong Wang model.">  <**ComponentType** name="Rwongwang"  description="Rate based 2D oscillator for TVB">  <**Constant** name="w\_plus" domain="" default="1.4f" description="Excitatory population recurrence weight"/>  <**Constant** name="a\_E" domain="" default="310.0f" description="[n/C]. Excitatory population input gain parameter, chosen to fit numerical solutions."/>  <**Constant** name="b\_E" domain="" default="125.0f" description="[Hz]. Excitatory population input shift parameter chosen to fit numerical solutions."/>  <**Constant** name="d\_E" domain="" default="0.154f" description="[s]. Excitatory population input scaling parameter chosen to fit numerical solutions."/>  <**Constant** name="a\_I" domain="" default="615.0f" description="[n/C]. Inhibitory population input gain parameter, chosen to fit numerical solutions."/>  <**Constant** name="b\_I" domain="" default="177.0f" description="[Hz]. Inhibitory population input shift parameter chosen to fit numerical solutions."/>  <**Constant** name="d\_I" domain="" default="0.087f" description="[s]. Inhibitory population input scaling parameter chosen to fit numerical solutions."/>  <**Constant** name="gamma\_E" domain="" default="0.641f / 1000.0f" description="Excitatory population kinetic parameter"/>  <**Constant** name="tau\_E" domain="" default="100.0f" description="[ms]. Excitatory population NMDA decay time constant."/>  <**Constant** name="tau\_I" domain="" default="10.0f" description="[ms]. Inhibitory population NMDA decay time constant."/>  <**Constant** name="I\_0" domain="" default="0.382f" description="[nA]. Effective external input"/>  <**Constant** name="w\_E" domain="" default="1.0f" description="Excitatory population external input scaling weight"/>  <**Constant** name="w\_I" domain="" default="0.7f" description="Inhibitory population external input scaling weight"/>  <**Constant** name="gamma\_I" domain="" default="1.0f / 1000.0f" description="Inhibitory population kinetic parameter"/>  <**Constant** name="min\_d\_E" domain="" default="-1.0f \* d\_E" description="Only in CUDA"/>  <**Constant** name="min\_d\_I" domain="" default="-1.0f \* d\_I" description="Only in CUDA"/>  <**Constant** name="imintau\_E" domain="" default="-1.0f / tau\_E" description="Only in CUDA"/>  <**Constant** name="imintau\_I" domain="" default="-1.0f / tau\_I" description="Only in CUDA"/>  <**Constant** name="w\_E\_\_I\_0" domain="" default="w\_E \* I\_0" description="Only in CUDA"/>  <**Constant** name="w\_I\_\_I\_0" domain="" default="w\_I \* I\_0" description="Only in CUDA"/>  <**Constant** name="J\_N" domain="" default="0.15" description="[nA] NMDA current"/>  <**Constant** name="J\_I" domain="" default="1.0" description="[nA] Local inhibitory current"/>  <**Constant** name="G" domain="" default="2.0" description="Global coupling scaling"/>  <**Constant** name="lamda" domain="" default="0.0" description="Inhibitory global coupling scaling"/>  <!-- Unpack params -->  <**Constant** name="J\_NMDA" domain="" default="0.15" description=""/>  <**Constant** name="JI" domain="" default="1.0" description=""/>  <**Constant** name="J\_NMDA" domain="" default="0.15" description=""/>  <**Constant** name="G\_J\_NMDA" domain="" default="G\*J\_NMDA" description=""/>  <**Constant** name="w\_plus\_\_J\_NMDA" domain="" default="w\_plus \* J\_NMDA" description=""/>  <!-- Parameters will be assumed to be swept. sweep data will be matched by order of entry -->  <**Parameter** name="global\_speed" dimension='float'/>  <**Parameter** name="global\_coupling" dimension='float'/>  <**DerivedParameter** name="rec\_n" expression="1 / n\_node"/>  <!-- If rec\_speed\_dt not is defined or is zero, then no delays will be fetched -->  <**DerivedParameter** name="rec\_speed\_dt" expression="0"/>  <**DerivedParameter** name="nsig" expression="sqrt(dt) \* sqrt(2.0 \* 1e-5)"/>  <**Dynamics**>  <**StateVariable** name="V" default="" boundaries="0.0000001, 1"/>  <**StateVariable** name="W" default="" boundaries="0.0000001, 1"/>  <**DerivedVariable** name="tmp\_I\_E" expression="a\_E \* (w\_E\_\_I\_0 + w\_plus\_\_J\_NMDA \* V + c\_0 - JI\*W) - b\_E"/>  <**DerivedVariable** name="tmp\_H\_E" expression="tmp\_I\_E/(1.0-exp(min\_d\_E \* tmp\_I\_E))"/>  <**DerivedVariable** name="tmp\_I\_I" expression="(a\_I\*((w\_I\_\_I\_0+(J\_NMDA \* V))-W))-b\_I"/>  <**DerivedVariable** name="tmp\_H\_I" expression="tmp\_I\_I/(1.0-exp(min\_d\_I\*tmp\_I\_I))"/>  <**TimeDerivative** name="dV" expression="(imintau\_E\* V)+(tmp\_H\_E\*(1-V)\*gamma\_E)"/>  <**TimeDerivative** name="dW" expression="(imintau\_I\* W)+(tmp\_H\_I\*gamma\_I)"/>  </**Dynamics**>  <!-- Exposures are used for observables, for the name enter variable to be observed (usually states)  and for dimension enter the reduction functionality-->  <**Exposure** name="V" choices="V" default="V"/>  </**ComponentType**>  <**ComponentType** name="coupling\_function">  <**Constant** name="a" domain="lo=0.0, hi=10., step=0.1" default="1" description="Rescales the connection strength."/>  <!-- variables for pre synaptic function-->  <**Parameter** name="V\_j" dimension='0'/>  <**Dynamics**>  <**DerivedVariable** name="pre" expression="V\_j \* G\_J\_NMDA" description = "pre synaptic function for coupling activity"/>  <**DerivedVariable** name="post" expression="a" description = "post synaptic function for coupling activity"/>  </**Dynamics**>  <!-- post coupling expression, needs to be present. If no temporary result needs to be computed enter 'None'-->  <**DerivedParameter** name="c\_0" expression="None"/>  </**ComponentType**>  <**ComponentType** name="noise" description="on"></**ComponentType**>  </**Lems**> |

1. We ignore higher-dimensional data at this point and just mention that -- if necessary at all -- it would be straightforward to serialize tensors into matrices. [↑](#footnote-ref-0)
2. One can either use the filename id as UUID (8 alphanumeric characters yield 2.8e+12 unique identifiers) or generate longer UUIDs that are stored inside the sidecar files. [↑](#footnote-ref-1)
3. https://github.com/the-virtual-brain/tvb-root/tree/master/scientific\_library/tvb/dsl [↑](#footnote-ref-2)
4. <http://lems.github.io/LEMS/> [↑](#footnote-ref-3)
5. http://lems.github.io/LEMS [↑](#footnote-ref-4)
6. Cannon et al., 2014, https://pubmed.ncbi.nlm.nih.gov/25309419/ [↑](#footnote-ref-5)
7. Vella et al., 2014, https://pubmed.ncbi.nlm.nih.gov/24795618/ [↑](#footnote-ref-6)
8. https://github.com/the-virtual-brain/tvb-root/tree/master/scientific\_library/tvb/dsl [↑](#footnote-ref-7)
9. https://github.com/the-virtual-brain/tvb-root/tree/master/scientific\_library/tvb/dsl\_cuda [↑](#footnote-ref-8)
10. http://lems.github.io/LEMS [↑](#footnote-ref-9)
11. Cannon et al., 2014, https://pubmed.ncbi.nlm.nih.gov/25309419/ [↑](#footnote-ref-10)
12. Vella et al., 2014, https://pubmed.ncbi.nlm.nih.gov/24795618/ [↑](#footnote-ref-11)
13. https://github.com/the-virtual-brain/tvb-root/tree/master/scientific\_library/tvb/dsl [↑](#footnote-ref-12)
14. https://github.com/the-virtual-brain/tvb-root/tree/master/scientific\_library/tvb/dsl\_cuda [↑](#footnote-ref-13)
15. Cannon et al., 2014, https://pubmed.ncbi.nlm.nih.gov/25309419/ [↑](#footnote-ref-14)
16. Vella et al., 2014, https://pubmed.ncbi.nlm.nih.gov/24795618/ [↑](#footnote-ref-15)
17. https://github.com/the-virtual-brain/tvb-root/tree/master/scientific\_library/tvb/dsl [↑](#footnote-ref-16)
18. https://github.com/the-virtual-brain/tvb-root/tree/master/scientific\_library/tvb/dsl\_cuda [↑](#footnote-ref-17)
19. https://github.com/the-virtual-brain/tvb-root/blob/master/scientific\_library/tvb/dsl\_cuda/XMLmodels/rwongwang\_CUDA.xml [↑](#footnote-ref-18)