BIDS Extension Proposal 28 (BEP028)
Provenance: BIDS-Prov

version 0.0.1 (draft)

This document is outdated, please see https://github.com/BEP028_BIDSprov/blob/master/bep028spec.md

/bids-standard

Extension moderators/leads: Satra Ghosh <satra@mit.edu> and Camille Maumet <camille.maumet@inria.fr>, Yaroslav O. Halchenko

Contributors: Stefan Appelhoff, Chris Markiewicz, Cyril R. Pernet, Jean-Baptiste Poline, Rémi Adon, Michael Dayan, Sarah Saneei, Eric Earl, Tibor Auer, Ghislain Vaillant, Matthieu Joulot, Omar El Rifai, Ryan J. Cali, Thomas Betton, Cyril Regan, Hermann Courteille, Arnaud Delorme.

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. 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.

Table of contents

1. Overview

1.1 Goals

Interpreting and comparing scientific results and enabling reusable data and analysis output require understanding provenance, i.e. how the data were generated and processed. To be useful, the provenance must be comprehensive, understandable, easily communicated, and captured automatically in machine accessible form.

This specification is aimed at describing the provenance of a BIDS dataset. This description is retrospective, i.e. it describes a set of steps that were executed in order to obtain the dataset (this is different from prospective descriptions of workflows that could for instance list all sets of steps that can be run on this dataset).

1.2 Which type of provenance is covered in this BEP?

Provenance comes up in many different contexts in BIDS. This specification focuses on representing the processings that were applied to a dataset. These could be for instance:

a. In BIDS derivatives, the description of which inputs from the BIDS dataset were used together, what software was run in what environment and with what parameters.
b. How a specific file in the BIDS dataset was generated.
c. In BIDS raw, conversion from DICOM images or other instrument native formats to BIDS layout.

BIDS-Prov is therefore currently limited to the capture of data processing, future considerations including other types of provenance are listed in section "Future perspectives".

1.3 File naming

This document describes the contents of a BIDS-Prov file; for naming and organization conventions, please consult the BIDS specification (https://bids-specification.readthedocs.io). Until these conventions are established in BIDS, it is RECOMMENDED to use the following:

BIDS-Prov files are JSON-LD files – i.e. a specific type of JSON files that allows encoding graph-like structures with the Resource Description Framework[1] –.

They can be stored in two different locations:

File level provenance. BIDS-Prov files can be stored immediately alongside any BIDS file (or BIDS-Derivatives file) they apply to. Each BIDS-Prov file must meet the following naming convention:

```
[sub-<label>/]
    [ses-<label>/]
        [<modality>/]
            <file-name-with-ext>.prov.jsonld
```

Dataset level provenance. BIDS-Prov files can be stored in a `prov/` directory immediately below the BIDS dataset (or BIDS-Derivatives dataset) root. Each BIDS-Prov file must meet the following naming convention, where label can be arbitrary.

```
prov/[<subfolders>*/]<label>.prov.jsonld
```

At the dataset level, provenance can be about any BIDS file in the dataset.

1.4 Top-level structure

BIDS-Prov JSON-LD file

A skeleton for a file-level or dataset-level BIDS-Prov JSON-LD file looks like this:

```
{
  "@context": "https://purl.org/nidash/bidsprov/context.json",
  "BIDSProvVersion": "0.0.1",
  "Records": {
        "Agents": [
        {
                <...Agent 1...>
        },
        {
                <...Agent 2...>
        }
        ],
        "Activities": [
        {
                <...Activity 1...>
                        <Used>
                        <AssociatedWith>
        }
        {
                <...Activity 2...>
        }
        ],
        "Entities": [
```

[1] https://www.w3.org/TR/json-ld11/#basic-concepts

```
        {
                <...Entity 1...>
                <GeneratedBy>
        },
        {
                <...Entity 2...>
        }
      ]
    }
  }
}
```

Here is a simple example:

```
{
"@context": "https://purl.org/nidash/bidsprov/context.json",
  "BIDSProvVersion": "0.0.1",
  "Records": {
    "Software": [
      {
        "Id": "urn:eeglab-4a586b50",
        "Label": "EEGLAB",
        "Version": "v2023"
      }
    ],
    "Activities": [
      {
        "Id": "urn:filter-00f3a18f",
        "Label": "Filter",
        "Used": "bids::sub-001/eeg/myfile_desc-filtered_eeg.set",
        "AssociatedWith": "urn:eeglab-4a586b50"
      }
    ],
    "Entities": [
      {
        "Id": "bids::sub-001/eeg/myfile_desc-filtered_eeg.set",
        "Label": "myfile_desc-filtered_eeg.set"
      },
      {
        "Id": "bids::sub-001/eeg/myfile_desc-filtered_downsampled_eeg.set",
        "Label": "myfile_desc-filtered_downsampled_eeg.set",
        "GeneratedBy": "urn:filter-00f3a18f"
      }
    ]
  }
}
```

Alternative representation for file-level provenance JSON-LD

Alternatively JSON-LD provenance information can look as follows where all the information is relative to the file it describes. Note that both representations are equivalent in RDF and can be used interchangeably at the discretion of the people writing up the provenance.

```
{
  "@context": "https://purl.org/nidash/bidsprov/context.json",
  "BIDSProvVersion": "0.0.1",
  <...Entity 1...>
  "GeneratedBy": {
    <...Activity...>
    "AssociatedWith": {
```

```
      <...Agent...>
    },
    "Used": {
      <...Entity 2...>
    }
  }
}
```

See a simple example here:

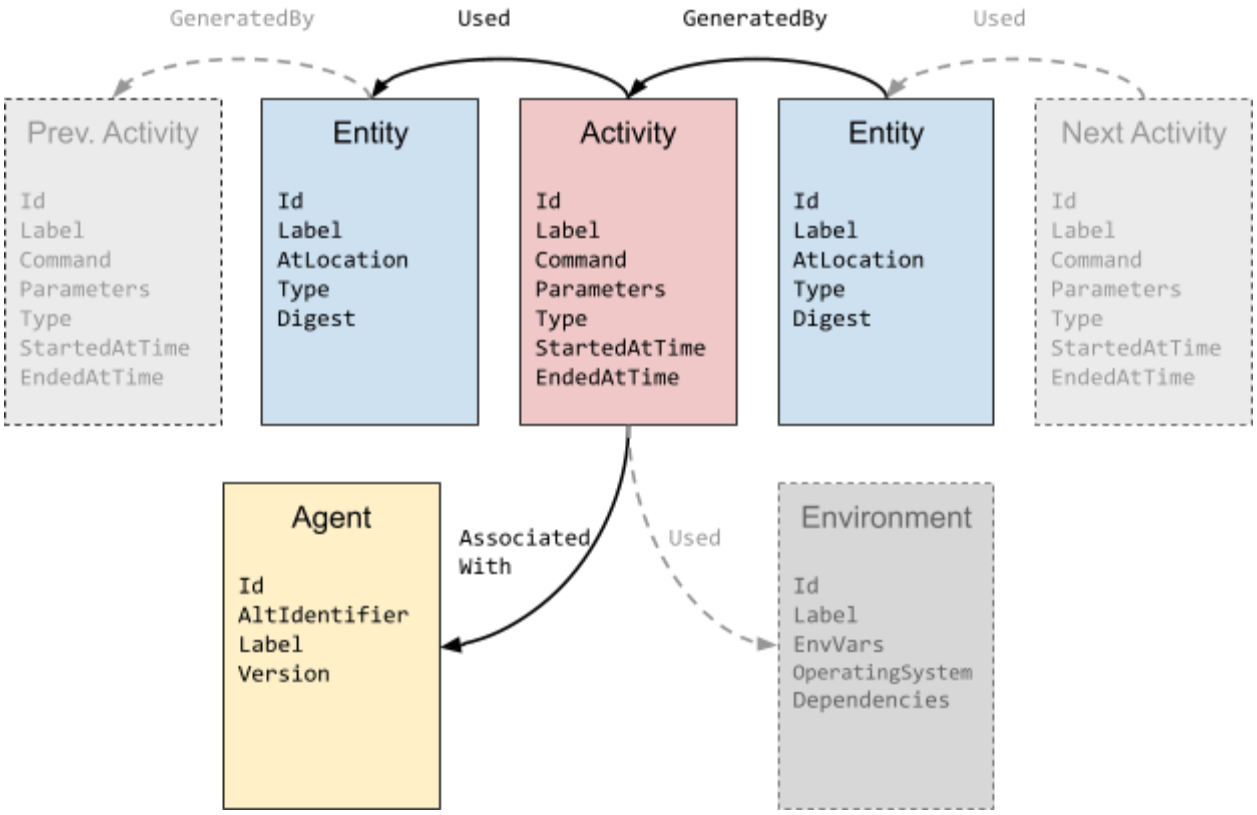
```
{
 "@context": "https://purl.org/nidash/bidsprov/context.json",
  "BIDSProvVersion": "0.0.1",
  "Records": {
        "Entities": [
        {
          "Id": "bids::sub-001/eeg/myfile_desc-filtered_eeg.set",
          "Label": "myfile_desc-filtered_eeg.set"
        },
        {
          "Id": "bids::sub-001/eeg/myfile_desc-filtered_downsampled_eeg.set",
          "Label": "myfile_desc-filtered_downsampled_eeg.set",
          "GeneratedBy":
          {
            "Id": "urn:filter-00f3a18f",
            "Type": "Activity",
            "Label": "Filter",
            "Used": "bids::sub-001/eeg/myfile_desc-filtered_eeg.set",
            "AssociatedWith":
            {
              "Id": "urn:eeglab-4a586b50",
              "Type": "Software",
              "Label": "EEGLAB",
              "Version": "v2023"
            }
        }
      }
     ]
  }
}
```

Key name	Description
`@context`	REQUIRED. A URL to the BIDS-Prov json context. Value must be "https://purl.org/nidash/bidsprov/context.json" (Note: this key-value pair is necessary so that the file can be read as a graph, see section "Graph model" for more details).
`BIDSProvVersion`	REQUIRED. A string identifying the version of the specification adhered to.
`Records`	REQUIRED. A list of Activity, Entity and Agent records describing the provenance (see "Activity", "Entity" and "Agent" sections below).

A complete schema for the model file to facilitate specification and validation is available from https://github.com/bids-standard/BEP028_BIDSprov. In the event of disagreements between the schema and the specification, the specification is authoritative.

2. Provenance records

Each provenance record is composed of a set of Activities that represent the transformations that have been applied to the data. Each Activity can have files (denoted as Entities) as inputs and outputs. The Agent specifies the software package.



2.1 Activity

Each Activity record has the following fields:

Key name	Description
`Id`	REQUIRED. Unique URIs (for example a UUID). Identifier for the activity.
`Label`	REQUIRED. String. Name of the tool, script, or function used (e.g. “bet”, "recon-all", "myFunc", "docker").
`Command`	REQUIRED. String. Command used to run the tool, including all parameters.
`Parameters`	OPTIONAL. Dict. A dictionary defining the parameters as key-value pairs.

`AssociatedWith`	OPTIONAL. UUID. Identifier of the software package used to compute this activity (the corresponding Agent must be defined with its own Agent record).
`Used`	OPTIONAL. UUID. Identifier of an entity used by this activity (the corresponding Entity must be defined with its own Entity record).
`Type`	OPTIONAL. URI. A term from a controlled vocabulary that more specifically describes the activity.
`StartedAtTime`	OPTIONAL. xsd:*dateTime.* A timestamp tracking when this activity started.
`EndedAtTime`	OPTIONAL. xsd:*dateTime.* A timestamp tracking when this activity ended

2.2 Entity

Each Entity (as a record or a top-level entity), a file, has the following fields:

Key name	**Description**
`Id`	REQUIRED. Unique URIs (for example a UUID). Identifier for the entity.
`Label`	REQUIRED. String. A name for the entity.
`AtLocation`	OPTIONAL. String. For input files, this is the relative path to the file on disk.
`GeneratedBy`	OPTIONAL. UUID. Identifier of the activity which generated this entity (the corresponding Activity must be defined with its own Activity record).
`Type`	OPTIONAL. URI. A term from a controlled vocabulary that more specifically describes the activity.
`Digest`	RECOMMENDED. Dict. For files, this would include checksums of files. It would take the form {"<checksum-name>": "value"}.

2.3 Agent (Optional)

Including an Agent record is OPTIONAL. If included, each Agent record has the following fields:

Key name	**Description**
`Id`	REQUIRED. A unique identifier like a UUID that will be used to associate activities with this software (e.g., urn:1264-1233-11231-12312, "urn:bet-o1ef4rt"

`AltIdentifier`	OPTIONAL. URI. For example, the RRID for this software package (cf. scicrunch).
`Label`	REQUIRED. String. Name of the software.
`Version`	REQUIRED. String. Version of the software.

2.4 Environments (Optional)

Information about the environment in which the provenance record was obtained is modeled with an environment record.

Environment records are OPTIONAL. If included, each environment record MUST have the following fields:

Key name	**Description**
`Id`	REQUIRED. Unique URIs (for example a UUID). Identifier for the environment (this identifier will be used to associated activities with this environment).
`Label`	REQUIRED. String. Name of the environment.
`EnvVars`	OPTIONAL. Dict. A dictionary defining the environment variables as key-value pairs.
`OperatingSystem`	OPTIONAL. String. Name of the operating system.
`Dependencies`	OPTIONAL. Dict. A dictionary defining the software used and their versions as key-value pairs.

3. Graph model

Note: since these jsonld documents are graph objects, they can be aggregated using RDF tools without the need to apply the inheritance principle.

4. Examples

A list of fMRI examples for BIDS-Prov are available for SPM, FSL and AFNI in: https://github.com/bids-standard/BEP028_BIDSprov/tree/master/examples

5. Future perspectives

Beyond what is covered in the current specification, provenance comes up in other contexts as well, which might be addressed at a later stage:

a. For datasets and derivatives, provenance can also include details of why the data were collected in the first place covering hypotheses, claims, and prior publications. Provenance can encode support for which claims were supported by future analyses.
b. Provenance can involve information about people and institutions involved in a study.
c. Provenance records can highlight reuse of datasets while providing appropriate attribution to the original dataset generators as well as future transformers.
d. Details of stimulus presentation and cognitive paradigms, and clinical and neuropsychiatric assessments, each come with their own details of provenance.
e. Transformations made by humans (e.g., editing freesurfer, adding quality evaluations).
f. The interpretability of provenance records requires a consistent vocabulary for provenance as well as an expectation for a consistent terminology for the objects being encoded. While the current specification focuses on the former, the latter (i.e. consistent terminology for the objects being encoded) will require additional efforts.