HED schemas

Release 0.0.1

HED Working Group

Dec 03, 2022

CONTENTS:

Downloads

HED tools

HED tools

Note: this is a work in progress. More information is coming. This is testing the update link.

CHAPTER

ONE

INTRODUCTION TO HED

This document contains the specification for third generation HED or HED-3G. It is meant for the implementers and users of HED tools. Other tutorials and tagging guides are available to researchers using HED to annotate their data. This document contains the specification for the first official release of HED-3G (HED versions 8.0.0-xxx and above.) **When the term HED is used in this document, it refers to third generation (HED-3G) unless explicitly stated otherwise.**

The aspects of HED that are described in this document are supported or will soon be supported by validators and other tools and are available for immediate use by annotators. The schema vocabulary can be viewed using an expandable schema viewer.

All HED-related source and documentation repositories are housed on the HED-standard organization GitHub site, https://github.com/hed-standard, which is maintained by the HED Working Group. HED development is open-source and community-based. Also see the official HED website https://www.hedtags.org for a list of additional resources.

The HED Working Group invites those interested in HED to contribute to the development process. Users are encouraged to use the *Issues* mechanism of the `hed-specification` repository on the GitHub `hed-standard` working group website: https://github.com/hed-standard/hed-specification/issues to ask for help or make suggestions. The HED discussion forum https://github.com/hed-standard/hed-specification/discussions is maintained for in depth discussions of HED issues and evolution.

Several other aspects of HED annotation are being planned, but their specification has not been fully determined. These aspects are not contained in this specification document, but rather are contained in ancillary working documents which are open for discussion. These ancillary specifications include the HED working document on spatial annotation and the HED working document on task annotation.

1.1 Scope of HED

HED (an acronym for Hierarchical Event Descriptors) is an evolving framework that facilitates the description and formal annotation of events identified in time series data, together with tools for validation and for using HED annotations in data search, extraction, and analysis. HED allows researchers to annotate what happened during an experiment, including experimental stimuli and other sensory events, participant responses and actions, experimental design, the role of events in the task, and the temporal structure of the experiment. The resulting annotation is machine-actionable, meaning that it can be used as input to algorithms without manual intervention. HED facilitates detailed comparisons of data across studies.

As the name HED implies, much of the HED framework focuses on associating metadata with the experimental timeline to make datasets analysis-ready and machine-actionable. However, HED annotations and framework can be used to incorporate other types of metadata into analysis by providing a common API (Application Programming Interface) for building inter-operable tools.

1.2 Role of library schemas

To avoid uncontrolled expansion of the base HED vocabulary with specialized terminology, HED supports the creation of library schema, which are specialized vocabularies that can be used in conjunction with the base schema and each other to annotate a variety of specialized

CHAPTER

TWO

LIBRARY SCHEMA OVERVIEW

The variety and complexity of events in electrophysiological experiments makes full documentation challenging. As more experiments move out of controlled laboratory environments and into less controlled virtual and real-world settings, the terminology required to adequately describe events has the potential to grow exponentially.

In addition, experiments in any given subfield can contribute to pressure to add overly-specific terms and jargon to the schema hierarchy—for example, adding musical terms to tag events in music-based experiments, video markup terms for experiments involving movie viewing, traffic terms for experiments involving virtual driving, and so forth.

Clinical fields using neuroimaging also have their own specific vocabularies for describing data features of clinical interest (e.g., seizure, sleep stage IV). Including these discipline-specific terms quickly makes the base HED schema unwieldy and less usable by the broader user community.

Third generation HED instead introduces the concept of the **HED library schema**. To use a programming analogy, when programmers write a Python module, the resulting code does not become part of the Python language or core library. Instead, the module becomes part of a library used in conjunction with core modules of the programming language.

Similar to the design principles imposed on function names and subclass organization in software development, HED library schemas must conform to some basic rules:

Rules for HED library schema design.

1. Every term must be unique within the library schema and must conform to the rules for HED schema terms.
2. Schema terms should be readily understood by most users. The terms should not be ambiguous and should be meaningful in themselves without reference to their position in the schema hierarchy.
3. If possible, no schema sub-tree should have more than 7 direct subordinate sub-trees.
4. Terms that are used independently of one another should be in different sub-trees (orthogonality).
5. Every term in the hierarchy satistifies the **is-a** relationship with its parent. In other words if B has A as a parent in the schema hierarchy, then B is an example of A. Searching for A will also return B (search generality).

As in Python programming, we anticipate that many HED schema libraries may be defined and used, in addition to the base HED schema. Libraries allow individual research or clinical communities to annotate details of events in experiments designed to answer questions of interest to particular to those communities.

Since it would be impossible to avoid naming conflicts across schema libraries that may be built in parallel by different user communities, HED supports schema library namespaces. Users will be able to add library tags qualified with namespace designators. All HED schemas, including library schemas, adhere to semantic versioning.

2.1 Defining a schema

A HED library schema is defined in the same way as the base HED schema except that it has an additional attribute name-value pair, `library="xxx"` in the schema header. We will use as an illustration a library schema for driving. Syntax details for a library schema are similar to those for the base HED schema. (See the HED schema format specification for more details).

Example: Driving library schema (MEDIAWIKI template).

```
HED library="driving" version="1.0.0"
!# start schema
   [... contents of the HED driving schema ...]
!# end schema
   [... required sections specifying schema attribute definitions ...]
!# end hed
```

The required sections specifying the schema attributes are *unit-class-specification*, *unit-modifier-specification*, *value-class-specification*, *schema-attribute-specification*, and *property-specification*.

Example: Driving library schema (XML template).

```
<?xml version="1.0" ?>
<HED library="driving" version="1.0.0">
    [... contents of the HED_DRIVE schema ... ]
</HED>
```

The schema XML file should be saved as `HED_driving_1.0.0.xml` to facilitate specification in tools.

2.2 Schema namespaces

As part of the HED annotation process, users must associate a standard HED schema with their datasets. Users may also include tags from an arbitrary number of additional library schemas. For each library schema used to annotate a data recording, the user must associate a local name with the appropriate library schema name and version. Each library must be associated with a distinct local name within a recording annotations. The local names should be strictly alphabetic with no blanks or punctuation.

The user must pass information about the library schema and their associated local names to processing functions. HED uses a standard method of identifying namespace elements by prefixing HED library schema tags with the associated local names. Tags from different library schemas can be intermixed with those of the base schema. Since the node names within a library must be unique, annotators can use short form as well as fully expanded tag paths for library schema tags as well as those from the base-schema.

Example: Driving library schema example tags.

```
dp:Action/Drive/Change-lanes
dp:Drive/Change-lanes
dp:Change-lanes
```

A colon (`:`) is used to separate the qualifying local name from the remainder of the tag. Notice that *Action* also appears in the standard HED schema. Identical terms may be used in a library schema and the standard HED schema. Use of the same term implies a similar purpose. Library schema developers should try not to reuse terms in the standard schema unless the intention is to convey a close or identical relationship.

2.3 Attributes and classes

In addition to the specification of tags in the main part of a schema, a HED schema has sections that specify unit classes, unit modifiers, value classes, schema attributes, and properties. The rules for the handling of these sections for a library schema are as follows:

2.3.1 Required sections

The required sections of a library schema are: the *schema-specification*, the *unit-class-specification*, the *unit-modifier-specification*, the *value-class-specification* section, the *schema-attribute-specification* section, and the *property-specification*. The library schema must include all required schema sections even if the content of these sections is empty.

2.3.2 Relation to base schema

Any schema attribute, unit class, unit modifier, value class, or property used in the library schema must be specified in the appropriate section of the library schema regardless of whether these appear in base schema. Validators check the library schema strictly on the basis of its own specification without reference to another schema.

2.3.3 Schema properties

HED only supports the schema properties listed in Table B.2: *boolProperty*, *unitClassProperty*, *unitModifierProperty*, *unitProperty*, and *valueClassProperty*.
If the library schema uses one of these in the library schema specification, then its specification must appear in the *property-specification* section of the library schema.

2.3.4 Unit classes

The library schema may define unit classes and units as desired or include unit classes or units from the base schema. Similarly, library schema may define unit modifiers or reuse unit modifiers from the base schema. HED validation and basic analysis tools validate these based strictly on the schema specification and do not use any outside information for these.

2.3.5 Value classes

The standard value classes (*dateTimeClass[]*, nameClass, numericClass[*]*, posixPath[]*, textClass[*]*) if used, should have the same meaning as in the base schema. The hard-coded behavior associated with the starred ([*]) value classes will be the same. Library schema may define additional value classes and specify their allowed characters, but no additional hard-coded behavior will be available in the standard toolset. This does not preclude special-purpose tools from incorporating their own behavior.

2.3.6 Schema attributes

The standard schema attributes (*allowedCharacter*, *defaultUnits*, *extensionAllowed*, *recommended*, *relatedTag*, *requireChild*, *required*, *SIUnit*, *SIUnitModifier*, *SIUnitSymbolModifier*, *suggestedTag*, *tagGroup*, *takesValue*, *topLevelTagGroup*, *unique*, *unitClass*, *unitPrefix*, *unitSymbol*, *valueClass*) should have the same meaning as in the base schema. The hard-coded behavior associated with the schema attributes will be the same. Library schema may define additional schema attributes. They will be checked for syntax, but no additional hard-coded behavior will be available in the standard toolset. This does not preclude special-purpose tools from incorporating their own behavior.

2.3.7 Syntax checking

Regardless of whether a specification is in the base-schema or not, HED tools can perform basic syntax checking.

Basic syntax checking for library schema.

1. All attributes used in the schema proper must be defined in the schema attribute section of the schema.
2. Undefined attributes cause an error in schema validation.
3. Similar rules apply to unit classes, unit modifiers, value classes, and properties.
4. Actual handling of the semantics by HED tools only occurs for entities appearing in the base schema.

2.4 Library schemas in BIDS

The most common use case (for 99.9% of the HED users) is to use one of the standard HED schemas available on GitHub in the `hedxml` directory of the `hed-specification` repository (https://github.com/hed-standard/hed-specification/tree/master/hedxml).

This section explains the changes that are being proposed in BIDS to accommodate access to HED library schemas. This section will be updated as the proposals progress though the BIDS review process. All `"fileName"` keys in the following discussion point to the names of files located in the `./code` directory of the dataset tree.

The major change to the BIDS specification is to allow the value associated with the `"HEDVersion"` key in the `dataset_description.json` file to be a dictionary rather than a string expressing the HED version. This proposed change will allow users more flexibility in specifying the base HED schema and will accommodate an arbitrary number of library schemas. The allowed top-level keys in this dictionary are: `"version"`, `"fileName"`, and `"libraries"`. The `"version"` and `"fileName"` keys pertain to the base HED schema, and if both are specified, `"version"` always takes precedence.

The `"libraries"` key points to a library dictionary. The keys of the library dictionary are the nicknames used in the dataset to reference the schema. The values The `"version"` key specifies the HED version number of a schema in the standard library and has the same effect as directly specifying. The following example specifies a base HED schema located in the `./code/myLocalSchema.xml` file of the dataset and two library schemas with nicknames `"la"` and `"lb"`. `

Example: Proposed specification of library schema in BIDS.

```
{
    "Name": "A wonderful experiment",
    "BIDSVersion": "1.4.0",
    "HEDVersion": {
```

(continues on next page)

(continued from previous page)

```
        "fileName": "mylocalSchema.xml",
        "libraries": {
            "la": {
                "libraryName": "libraryA",
                "version": "1.0.2"
            },
            "lb": {
                "fileName": "HED_libraryB_0.5.3.xml"
            }
        }
    }
}
```

The "la" library schema is the `./hedxml/HED_libraryA_1.0.2.xml` file found in the hed-schema-library repository on the hed-standard working group GitHub site. HED tags from this library have the `la:` prefix (e.g., `la:XXX`). The "lb" library schema can be found in the `./code/HED_libraryB_0.5.3.xml` file in the BIDS dataset. Tags from this library are prefixed with `lb:`. NOTE: This specification is preliminary and is waiting the resolution of BIDS formats for specifying external files as outline in BIDS specification PR #820.

CHAPTER

THREE

SCORE SCHEMA LIBRARY

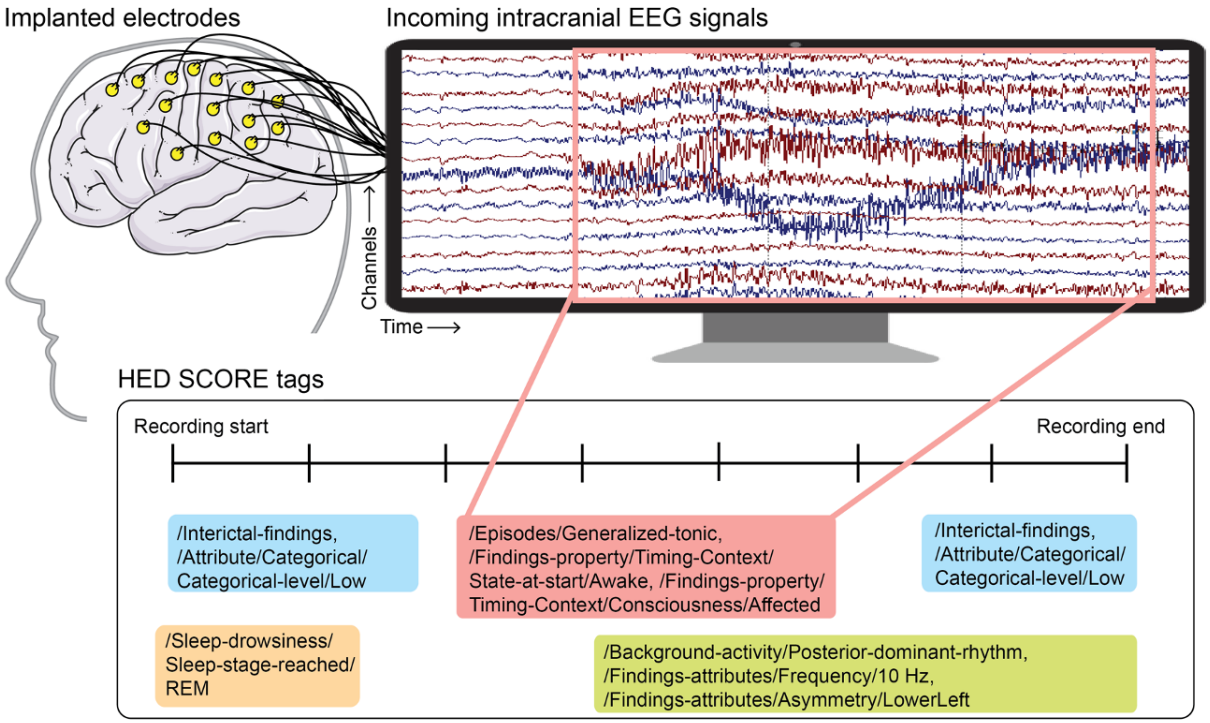
The manual review and interpretation of the EEG is critical to the value of the EEG as a diagnostic tool but is highly sensitive to subjectivity. The consensus between experts interpreting the same EEG is higher when using the same structured standard. Still, EEG is very complex, making it difficult to describe it in structured fields adequately. Free-text fields might be more flexible but suffer from incompleteness, lack of consistency, and the limited ability to share reports. Therefore, the standardization of clinical terminology holds considerable potential for the field.

The Standardized Computer-based Organized Reporting of EEG (SCORE)[1]^,^[2] is a standardized terminology for annotating EEG and ictal clinical events currently used in commercial software to create a standardized report. The SCORE standard goal is to give epileptologists a computerized tool and standard terminology that can be used in clinical practice and maximize interobserver agreement. The 1^st^ SCORE version received European consensus, endorsed by the European Chapter of the IFCN and the International League Against Epilepsy (ILAE) Commission on European Affairs. The 2^nd^ SCORE version is a revised terminology extended from the 1^st^ version with additional terms from multiple classifications, glossaries, and standard terminologies that have achieved international consensus.

The SCORE implementation in HED tackles these textual reports' lack of machine readability and makes SCORE available for and machine-readable by open-source software. Many researchers worldwide can use the SCORE library to reduce errors in clinical evaluations and perform advanced mega-analyses, ultimately advancing the understanding of the human brain.

[1] Beniczky, Sándor, et al. "Standardized computer-based organized reporting of EEG: SCORE." Epilepsia 54.6 (2013): 1112-1124.

[2] Beniczky, Sándor, et al. "Standardized computer-based organized reporting of EEG: SCORE–second version." Clinical Neurophysiology 128.11 (2017): 2334-2346.



3.1 Development

The SCORE HED schema library maintains the hierarchy as in the SCORE EEG Educational Platform[3] and presented in SCORE papers. The SCORE EEG Educational Platform is interactive web-based software that teaches how to use the SCORE standard. The educational platform guided the development process by permitting the interactive inspection of the main types of EEG graphoelements included in the SCORE report and the appropriate low-level nodes for each of them.

The top levels of the HED schema library for SCORE correspond to the main EEG graphoelements, including Modulators, Background activity, Sleep and drowsiness, Interictal findings, Episodes, Physiologic patterns, Uncertain significance patterns, EEG artifacts, and Polygraphic channels.

The SCORE HED schema library is intended to describe all normal and abnormal EEG features. Therefore, description of patient information, referral and recording condition information, administrative data, and continuous EEG monitoring in neonates is beyond this scope.

3.1.1 Validation

The HED schema library for SCORE was converted and validated using the HED tools. See more here.

[3] https://www.holbergeeg.com/educational-platform

3.2 Brain imaging data structure (BIDS)

HED schema library for SCORE is compatible with the BIDS human and machine-readable events annotations .tsv files, see more here. A HED schema library for SCORE annotations implementation example is available here.

Manuscript: Tal Pal Attia et al., (in prep). "Hierarchical Event Descriptor library schema for clinical EEG data annotation".

CHAPTER

FOUR

DOCUMENTATION

4.1 1. HED publications

Explanation of the history, development, and motivation for third generation HED:

> Robbins, K., Truong, D., Jones, A., Callanan, I., & Makeig, S. (2020, August 1). Building FAIR functionality: Annotating events in time series data using Hierarchical Event Descriptors (HED). https://doi.org/10.31219/osf.io/5fg73

Detailed case study in using HED-3G for tagging:

> Robbins, K., Truong, D., Appelhoff, S., Delorme, A., & Makeig, S. (2021, May 7). Capturing the nature of events and event context using Hierarchical Event Descriptors (HED). BioRxiv, 2021.05.06.442841. https://doi.org/10.1101/2021.05.06.442841

4.2 2. Working documents

Mapping of HED terms and their descriptions to known ontologies is:

> HED-3G Working Document on Ontology mapping https://drive.google.com/file/d/13y17OwwNBlHdhB7hguSmOBdxn0Uk4hsI/view?usp=sharing

Two other working documents hold portions of the HED-3G specification that are under development and will not be finalized for Release 1:

> HED-3G Working Document on Spatial Annotation https://docs.google.com/document/d/1jpSASpWQwOKtan15iQeiYHVewvEeefcBUn1xipNH5-8/view?usp=sharing

> HED-3G Working Document on Task Annotation https://docs.google.com/document/d/1eGRI_gkYutmwmAl524ezwkX7VwikrLTQa9t8PocQMlU/view?usp=sharing

4.3 3. Schema viewers

The HED schema is usually developed in `.mediawiki` format and converted to XML for use by tools. However, researchers wishing to tag datasets will find both of these views hard to read. For this reason, we provide links to three versions of the schema. The expandable HTML viewer is easier to navigate. Annotators can also use CTAGGER, which includes a schema viewer and tagging hints.

Table 1: HED web-based schema vocabulary viewers.

Viewer	Link
Expandable HTML	https://www.hedtags.org/display_hed.html?version=8.0.0
Mediawiki	https://github.com/hed-standard/hed-specification/blob/master/hedwiki/HED8.0.0.mediawiki
XML	https://github.com/hed-standard/hed-specification/blob/master/hedxml/HED8.0.0.xml

4.4 4. HED Websites

The following is a summary of the HED-related websites

Table 2: HED websites.

Description	Site
Information and documentation	
HED organization website	https://www.hedtags.org
HED organization github	https://github.com/hed-standard
HED specification repository	https://github.com/hed-standard/hed-specification
Examples of HED annotation	https://github.com/hed-standard/hed-examples
HED documentation website	https://github.com/hed-standard/hed-standard.github.io
HED Python resources	
Python code repository	https://github.com/hed-standard/hed-python
Python validator and tools	https://github.com/hed-standard/hed-python/tree/master/hedtools
HED JavaScript resources	
HED JavaScript code	https://github.com/hed-standard/hed-javascript
BIDS validator	https://github.com/bids-standard/bids-validator
HED Matlab resources	
Matlab source code	https://github.com/hed-standard/hed-matlab
Annotator resources	
CTAGGER executable jar	https://github.com/hed-standard/hed-java/raw/master/ctagger.jar
CTAGGER repository	https://github.com/hed-standard/CTagger
Java repository	https://github.com/hed-standard/hed-java
Online HED tools	
Online website	https://hedtools.ucsd.edu/hed
Docker deployment	https://github.com/hed-standard/hed-python/tree/master/webtools/deploy_hed