# HED Python

*Release 0.3.1*

**HED Working Group**

**Sep 05, 2023**

# CONTENTS:


**Links**

- PDF docs
- Source code

Note: this is a work in progress. More information is coming.

CHAPTER

ONE

# INTRODUCTION TO HED

**Contents**



## 1.1 Why HED?

**Why use HED?**

HED (Hierarchical Event Descriptors) is an infrastructure and a controlled vocabulary that allows researchers to annotate their experimental data, especially events, so that tools can automatically use this information in analysis.

For more information on using Hierarchical Event Descriptors (HED) visit HED examples:

## 1.2 Installing hedtools

Hedtools will be available soon on pypi, but in the meantime, you can install directly from the GitHub repository using the following command:

```
`code >>> pip install git+https://github.com/hed-standard/hed-python.git `
```

## 1.3 Finding help

**Documentation**

See HED resources for user documentation and tutorials.

The HED online tools provide an easy-to-use interface that requires no programming.

**Mailing lists and forums**

- Don't hesitate to ask questions about the python hedtools on NeuroStars.

**Issues and problems**

- If you notice a bug in the python hedtools code or encounter other problems using the tools, please open an issue in the hed-python repository on github.

CHAPTER

TWO

# HED TOOLS USER GUIDE

CHAPTER

THREE

# HED API REFERENCE (AUTO STYLE)

| | |
|---|---|
| *errors* | |
| *models* | Data structures for HED tag handling. |
| *schema* | Data structures for handling the HED schema. |
| *tools* | HED remodeling, analysis and summarization tools. |
| *validator* | Validation of HED tags. |

## 3.1 hed.errors

### Modules

| | |
|---|---|
| *hed.errors.error_messages* | This module contains the actual formatted error messages for each type. |
| *hed.errors.error_reporter* | This module is used to report errors found in the validation. |
| *hed.errors.error_types* | |
| *hed.errors.exceptions* | |
| *hed.errors.known_error_codes* | |
| *hed.errors.schema_error_messages* | |

### 3.1.1 hed.errors.error_messages

This module contains the actual formatted error messages for each type.

Add new errors here, or any other file imported after error_reporter.py.

### Functions

| | |
|---|---|
| *SIDECAR_HED_USED*() | |
| *SIDECAR_HED_USED_COLUMN*() | |
| *def_error_bad_location*(tag) | |
| *def_error_def_tag_in_definition*(tag, def_name) | |
| *def_error_duplicate_definition*(def_name) | |
| *def_error_invalid_def_extension*(tag, def_name) | |
| *def_error_no_group_tags*(def_name) | |
| *def_error_no_takes_value*(tag, def_name) | |
| *def_error_wrong_group_tags*(def_name, tag_list) | |
| *def_error_wrong_placeholder_count*(def_name, ...) | |
| *invalid_column_ref*(bad_ref) | |
| *nested_column_ref*(column_name, index, symbol) | |
| *onset_error_def_unmatched*(tag) | |
| *onset_error_inset_before_onset*(tag) | |
| *onset_error_offset_before_onset*(tag) | |
| *onset_error_same_defs_one_row*(tag, def_name) | |
| *onset_no_def_found*(tag) | |
| *onset_too_many_defs*(tag, tag_list) | |
| *onset_too_many_groups*(tag, tag_list) | |
| *onset_wrong_placeholder*(tag, has_placeholder) | |
| *onset_wrong_type_tag*(tag, def_tag) | |
| *self_column_ref*(self_ref) | |
| *sidecar_error_blank_hed_string*() | |
| *sidecar_error_hed_data_type*(expected_type, ...) | |

continues on next page

Table 1 – continued from previous page

| | |
|---|---|
| *sidecar_error_invalid_pound_sign_count*(...) | |
| *sidecar_error_too_many_pound_signs*(...) | |
| *sidecar_error_unknown_column*(column_name) | |
| *sidecar_na_used*(column_name) | |
| *val_error_bad_def_expand*(tag, actual_def, ...) | |
| *val_error_comma_missing*(tag) | |
| *val_error_def_expand_unmatched*(tag) | |
| *val_error_def_expand_value_extra*(tag) | |
| *val_error_def_expand_value_missing*(tag) | |
| *val_error_def_unmatched*(tag) | |
| *val_error_def_value_extra*(tag) | |
| *val_error_def_value_missing*(tag) | |
| *val_error_duplicate_clumn*(column_number, ...) | |
| *val_error_duplicate_group*(group) | |
| *val_error_duplicate_tag*(tag) | |
| *val_error_empty_group*(tag) | |
| *val_error_extra_column*(column_name) | |
| *val_error_extra_comma*(source_string, char_index) | |
| *val_error_extra_slashes_spaces*(tag, problem_tag) | |
| *val_error_hed_blank_column*(column_number) | |
| *val_error_invalid_char*(source_string, char_index) | |
| *val_error_invalid_extension*(tag) | |
| *val_error_invalid_parent*(tag, problem_tag, ...) | |
| *val_error_invalid_tag_character*(tag, problem_tag) | |
| *val_error_invalid_unit*(tag, units) | |

continues on next page

Table 1 – continued from previous page

| | |
|---|---|
| *val_error_missing_column*(column_name, ...) | |
| *val_error_multiple_unique*(tag_namespace) | |
| *val_error_no_valid_tag*(tag, problem_tag) | |
| *val_error_no_value*(tag) | |
| *val_error_parentheses*(...) | |
| *val_error_prefix_invalid*(tag, tag_namespace) | |
| *val_error_require_child*(tag) | |
| *val_error_sidecar_key_missing*(invalid_key, ...) | |
| *val_error_sidecar_with_column*(column_names) | |
| *val_error_tag_extended*(tag, problem_tag) | |
| *val_error_tag_group_tag*(tag) | |
| *val_error_tildes_not_supported*(...) | |
| *val_error_top_level_tag*(tag) | |
| *val_error_top_level_tags*(tag, multiple_tags) | |
| *val_error_unknown_namespace*(tag, ...) | |
| *val_warning_capitalization*(tag) | |
| *val_warning_default_units_used*(tag, default_unit) | |
| *val_warning_required_prefix_missing*(...) | |

#### 3.1.1.1 hed.errors.error_messages.SIDECAR_HED_USED

**SIDECAR_HED_USED**()

#### 3.1.1.2 hed.errors.error_messages.SIDECAR_HED_USED_COLUMN

SIDECAR_HED_USED_COLUMN()

#### 3.1.1.3 hed.errors.error_messages.def_error_bad_location

def_error_bad_location(*tag*)

#### 3.1.1.4 hed.errors.error_messages.def_error_def_tag_in_definition

def_error_def_tag_in_definition(*tag*, *def_name*)

#### 3.1.1.5 hed.errors.error_messages.def_error_duplicate_definition

def_error_duplicate_definition(*def_name*)

#### 3.1.1.6 hed.errors.error_messages.def_error_invalid_def_extension

def_error_invalid_def_extension(*tag*, *def_name*)

#### 3.1.1.7 hed.errors.error_messages.def_error_no_group_tags

def_error_no_group_tags(*def_name*)

#### 3.1.1.8 hed.errors.error_messages.def_error_no_takes_value

def_error_no_takes_value(*tag*, *def_name*)

#### 3.1.1.9 hed.errors.error_messages.def_error_wrong_group_tags

def_error_wrong_group_tags(*def_name*, *tag_list*)

#### 3.1.1.10 hed.errors.error_messages.def_error_wrong_placeholder_count

def_error_wrong_placeholder_count(*def_name*, *expected_count*, *tag_list*)

#### 3.1.1.11 hed.errors.error_messages.invalid_column_ref

invalid_column_ref(*bad_ref*)

#### 3.1.1.12 hed.errors.error_messages.nested_column_ref

**nested_column_ref**(*column_name*, *index*, *symbol*)

#### 3.1.1.13 hed.errors.error_messages.onset_error_def_unmatched

**onset_error_def_unmatched**(*tag*)

#### 3.1.1.14 hed.errors.error_messages.onset_error_inset_before_onset

**onset_error_inset_before_onset**(*tag*)

#### 3.1.1.15 hed.errors.error_messages.onset_error_offset_before_onset

**onset_error_offset_before_onset**(*tag*)

#### 3.1.1.16 hed.errors.error_messages.onset_error_same_defs_one_row

**onset_error_same_defs_one_row**(*tag*, *def_name*)

#### 3.1.1.17 hed.errors.error_messages.onset_no_def_found

**onset_no_def_found**(*tag*)

#### 3.1.1.18 hed.errors.error_messages.onset_too_many_defs

**onset_too_many_defs**(*tag*, *tag_list*)

#### 3.1.1.19 hed.errors.error_messages.onset_too_many_groups

**onset_too_many_groups**(*tag*, *tag_list*)

#### 3.1.1.20 hed.errors.error_messages.onset_wrong_placeholder

**onset_wrong_placeholder**(*tag*, *has_placeholder*)

#### 3.1.1.21 hed.errors.error_messages.onset_wrong_type_tag

**onset_wrong_type_tag**(*tag*, *def_tag*)

#### 3.1.1.22 hed.errors.error_messages.self_column_ref

**self_column_ref**(*self_ref*)

#### 3.1.1.23 hed.errors.error_messages.sidecar_error_blank_hed_string

**sidecar_error_blank_hed_string**()

#### 3.1.1.24 hed.errors.error_messages.sidecar_error_hed_data_type

**sidecar_error_hed_data_type**(*expected_type*, *given_type*)

#### 3.1.1.25 hed.errors.error_messages.sidecar_error_invalid_pound_sign_count

**sidecar_error_invalid_pound_sign_count**(*pound_sign_count*)

#### 3.1.1.26 hed.errors.error_messages.sidecar_error_too_many_pound_signs

**sidecar_error_too_many_pound_signs**(*pound_sign_count*)

#### 3.1.1.27 hed.errors.error_messages.sidecar_error_unknown_column

**sidecar_error_unknown_column**(*column_name*)

#### 3.1.1.28 hed.errors.error_messages.sidecar_na_used

**sidecar_na_used**(*column_name*)

#### 3.1.1.29 hed.errors.error_messages.val_error_bad_def_expand

**val_error_bad_def_expand**(*tag*, *actual_def*, *found_def*)

#### 3.1.1.30 hed.errors.error_messages.val_error_comma_missing

**val_error_comma_missing**(*tag*)

#### 3.1.1.31 hed.errors.error_messages.val_error_def_expand_unmatched

**val_error_def_expand_unmatched**(*tag*)

#### 3.1.1.32 hed.errors.error_messages.val_error_def_expand_value_extra

val_error_def_expand_value_extra(*tag*)

#### 3.1.1.33 hed.errors.error_messages.val_error_def_expand_value_missing

val_error_def_expand_value_missing(*tag*)

#### 3.1.1.34 hed.errors.error_messages.val_error_def_unmatched

val_error_def_unmatched(*tag*)

#### 3.1.1.35 hed.errors.error_messages.val_error_def_value_extra

val_error_def_value_extra(*tag*)

#### 3.1.1.36 hed.errors.error_messages.val_error_def_value_missing

val_error_def_value_missing(*tag*)

#### 3.1.1.37 hed.errors.error_messages.val_error_duplicate_clumn

val_error_duplicate_clumn(*column_number*, *column_name*, *list_names*)

#### 3.1.1.38 hed.errors.error_messages.val_error_duplicate_group

val_error_duplicate_group(*group*)

#### 3.1.1.39 hed.errors.error_messages.val_error_duplicate_tag

val_error_duplicate_tag(*tag*)

#### 3.1.1.40 hed.errors.error_messages.val_error_empty_group

val_error_empty_group(*tag*)

#### 3.1.1.41 hed.errors.error_messages.val_error_extra_column

val_error_extra_column(*column_name*)

#### 3.1.1.42 hed.errors.error_messages.val_error_extra_comma

**val_error_extra_comma**(*source_string*, *char_index*)

#### 3.1.1.43 hed.errors.error_messages.val_error_extra_slashes_spaces

**val_error_extra_slashes_spaces**(*tag*, *problem_tag*)

#### 3.1.1.44 hed.errors.error_messages.val_error_hed_blank_column

**val_error_hed_blank_column**(*column_number*)

#### 3.1.1.45 hed.errors.error_messages.val_error_invalid_char

**val_error_invalid_char**(*source_string*, *char_index*)

#### 3.1.1.46 hed.errors.error_messages.val_error_invalid_extension

**val_error_invalid_extension**(*tag*)

#### 3.1.1.47 hed.errors.error_messages.val_error_invalid_parent

**val_error_invalid_parent**(*tag*, *problem_tag*, *expected_parent_tag*)

#### 3.1.1.48 hed.errors.error_messages.val_error_invalid_tag_character

**val_error_invalid_tag_character**(*tag*, *problem_tag*)

#### 3.1.1.49 hed.errors.error_messages.val_error_invalid_unit

**val_error_invalid_unit**(*tag*, *units*)

#### 3.1.1.50 hed.errors.error_messages.val_error_missing_column

**val_error_missing_column**(*column_name*, *column_type*)

#### 3.1.1.51 hed.errors.error_messages.val_error_multiple_unique

**val_error_multiple_unique**(*tag_namespace*)

#### 3.1.1.52 hed.errors.error_messages.val_error_no_valid_tag

**val_error_no_valid_tag**(*tag*, *problem_tag*)

#### 3.1.1.53 hed.errors.error_messages.val_error_no_value

**val_error_no_value**(*tag*)

#### 3.1.1.54 hed.errors.error_messages.val_error_parentheses

**val_error_parentheses**(*opening_parentheses_count*, *closing_parentheses_count*)

#### 3.1.1.55 hed.errors.error_messages.val_error_prefix_invalid

**val_error_prefix_invalid**(*tag*, *tag_namespace*)

#### 3.1.1.56 hed.errors.error_messages.val_error_require_child

**val_error_require_child**(*tag*)

#### 3.1.1.57 hed.errors.error_messages.val_error_sidecar_key_missing

**val_error_sidecar_key_missing**(*invalid_key*, *category_keys*)

#### 3.1.1.58 hed.errors.error_messages.val_error_sidecar_with_column

**val_error_sidecar_with_column**(*column_names*)

#### 3.1.1.59 hed.errors.error_messages.val_error_tag_extended

**val_error_tag_extended**(*tag*, *problem_tag*)

#### 3.1.1.60 hed.errors.error_messages.val_error_tag_group_tag

**val_error_tag_group_tag**(*tag*)

#### 3.1.1.61 hed.errors.error_messages.val_error_tildes_not_supported

**val_error_tildes_not_supported**(*source_string*, *char_index*)

#### 3.1.1.62 hed.errors.error_messages.val_error_top_level_tag

**val_error_top_level_tag**(*tag*)

#### 3.1.1.63 hed.errors.error_messages.val_error_top_level_tags

**val_error_top_level_tags**(*tag*, *multiple_tags*)

#### 3.1.1.64 hed.errors.error_messages.val_error_unknown_namespace

**val_error_unknown_namespace**(*tag*, *unknown_prefix*, *known_prefixes*)

#### 3.1.1.65 hed.errors.error_messages.val_warning_capitalization

**val_warning_capitalization**(*tag*)

#### 3.1.1.66 hed.errors.error_messages.val_warning_default_units_used

**val_warning_default_units_used**(*tag*, *default_unit*)

#### 3.1.1.67 hed.errors.error_messages.val_warning_required_prefix_missing

**val_warning_required_prefix_missing**(*tag_namespace*)

### 3.1.2 hed.errors.error_reporter

This module is used to report errors found in the validation.

You can scope the formatted errors with calls to push_error_context and pop_error_context.

#### Functions

| | |
|---|---|
| *check_for_any_errors*(issues_list) | Returns True if there are any errors with a severity of warning |
| *create_doc_link*(error_code) | If error code is a known code, return a documentation url for it |
| *get_printable_issue_string*(issues[, title, ...]) | Return a string with issues list flatted into single string, one per line. |
| *get_printable_issue_string_html*(issues[, ...]) | Return a string with issues list as an HTML tree. |
| *hed_error*(error_type[, default_severity, ...]) | Decorator for errors in error handler or inherited classes. |
| *hed_tag_error*(error_type[, ...]) | Decorator for errors in error handler or inherited classes. |
| *replace_tag_references*(list_or_dict) | Utility function to remove any references to tags, strings, etc from any type of nested list or dict |
| *sort_issues*(issues[, reverse]) | Sorts a list of issues by the error context values. |

#### 3.1.2.1 hed.errors.error_reporter.check_for_any_errors

**check_for_any_errors**(*issues_list*)

Returns True if there are any errors with a severity of warning

#### 3.1.2.2 hed.errors.error_reporter.create_doc_link

**create_doc_link**(*error_code*)

If error code is a known code, return a documentation url for it

**Parameters**
**error_code** (*str*) – A HED error code

**Returns**
The URL if it's a valid code

**Return type**
url(str or None)

#### 3.1.2.3 hed.errors.error_reporter.get_printable_issue_string

**get_printable_issue_string**(*issues*, *title=None*, *severity=None*, *skip_filename=True*, *add_link=False*)

Return a string with issues list flatted into single string, one per line.

**Parameters**

- **issues** (*list*) – Issues to print.
- **title** (*str*) – Optional title that will always show up first if present(even if there are no validation issues).
- **severity** (*int*) – Return only warnings >= severity.
- **skip_filename** (*bool*) – If true, don't add the filename context to the printable string.
- **add_link** (*bool*) – Add a link at the end of message to the appropriate error if True

**Returns**
A string containing printable version of the issues or ''.

**Return type**
str

#### 3.1.2.4 hed.errors.error_reporter.get_printable_issue_string_html

**get_printable_issue_string_html**(*issues*, *title=None*, *severity=None*, *skip_filename=True*)

Return a string with issues list as an HTML tree.

**Parameters**

- **issues** (*list*) – Issues to print.
- **title** (*str*) – Optional title that will always show up first if present.
- **severity** (*int*) – Return only warnings >= severity.
- **skip_filename** (*bool*) – If true, don't add the filename context to the printable string.

**Returns**
An HTML string containing the issues or ‘’.

**Return type**
str

#### 3.1.2.5 hed.errors.error_reporter.hed_error

**hed_error**(*error_type*, *default_severity=1*, *actual_code=None*)

Decorator for errors in error handler or inherited classes.

**Parameters**

- **error_type** (*str*) – A value from error_types or optionally another value.
- **default_severity** (ErrorSeverity) – The default severity for the decorated error.
- **actual_code** (*str*) – The actual error to report to the outside world.

#### 3.1.2.6 hed.errors.error_reporter.hed_tag_error

**hed_tag_error**(*error_type*, *default_severity=1*, *has_sub_tag=False*, *actual_code=None*)

Decorator for errors in error handler or inherited classes.

**Parameters**

- **error_type** (*str*) – A value from error_types or optionally another value.
- **default_severity** (ErrorSeverity) – The default severity for the decorated error.
- **has_sub_tag** (*bool*) – If true, this error message also wants a sub_tag passed down. eg “This” in “This/Is/A/Tag”
- **actual_code** (*str*) – The actual error to report to the outside world.

#### 3.1.2.7 hed.errors.error_reporter.replace_tag_references

**replace_tag_references**(*list_or_dict*)

Utility function to remove any references to tags, strings, etc from any type of nested list or dict

Use this if you want to save out issues to a file.

If you’d prefer a copy returned, use replace_tag_references(list_or_dict.copy())

**Parameters**
**list_or_dict** (*list or dict*) – An arbitrarily nested list/dict structure

#### 3.1.2.8 hed.errors.error_reporter.sort_issues

**sort_issues**(*issues*, *reverse=False*)

Sorts a list of issues by the error context values.

**Parameters**

- **issues** (*list*) – A list of dictionaries representing the issues to be sorted.
- **reverse** (*bool, optional*) – If True, sorts the list in descending order. Default is False.

**Returns**

The sorted list of issues.

**Return type**

list

### Classes

| |
|---|
| *ErrorHandler*([check_for_warnings]) |

#### 3.1.2.9 hed.errors.error_reporter.ErrorHandler

**class ErrorHandler**(*check_for_warnings=True*)

Bases: object

**__init__**(*check_for_warnings=True*)

### Methods

| | |
|---|---|
| *__init__*([check_for_warnings]) | |
| *add_context_and_filter*(issues) | Filter out warnings if requested, while adding context to issues. |
| *filter_issues_by_severity*(issues_list, severity) | Gather all issues matching or below a given severity. |
| *format_error*(error_type, *args[, actual_error]) | Format an error based on the parameters, which vary based on what type of error this is. |
| *format_error_from_context*(error_type, ...[, ...]) | Format an error based on the error type. |
| format_error_with_context(*args, **kwargs) | |
| get_error_context_copy() | |
| *pop_error_context*() | Remove the last scope from the error context. |
| *push_error_context*(context_type, context) | Push a new error context to narrow down error scope. |
| *reset_error_context*() | Reset all error context information to defaults. |
| *val_error_unknown*(**kwargs) | Default error handler if no error of this type was registered. |

add_context_and_filter(*issues*)

Filter out warnings if requested, while adding context to issues.

**issues(list):**
list: A list containing a single dictionary representing a single error.

static filter_issues_by_severity(*issues_list*, *severity*)

Gather all issues matching or below a given severity.

**Parameters**

- **issues_list** (*list*) – A list of dictionaries containing the full issue list.
- **severity** (*int*) – The level of issues to keep.

**Returns**
A list of dictionaries containing the issue list after filtering by severity.

**Return type**
list

static format_error(*error_type*, **args*, *actual_error=None*, ***kwargs*)

Format an error based on the parameters, which vary based on what type of error this is.

**Parameters**

- **error_type** (*str*) – The type of error for this. Registered with @hed_error or @hed_tag_error.
- **args** (*args*) – Any remaining non keyword args after those required by the error type.
- **actual_error** (*str or None*) – Code to actually add to report out.
- **kwargs** (*kwargs*) – The other keyword args to pass down to the error handling func.

**Returns**
A list containing a single dictionary representing a single error.

**Return type**
list

### Notes

The actual error is useful for errors that are shared like invalid character.

static format_error_from_context(*error_type*, *error_context*, **args*, *actual_error=None*, ***kwargs*)

Format an error based on the error type.

**Parameters**

- **error_type** (*str*) – The type of error. Registered with @hed_error or @hed_tag_error.
- **error_context** (*list*) – Contains the error context to use for this error.
- **args** (*args*) – Any remaining non keyword args.
- **actual_error** (*str or None*) – Error code to actually add to report out.
- **kwargs** (*kwargs*) – Keyword parameters to pass down to the error handling func.

**Returns**
A list containing a single dictionary

**Return type**
list

### Notes

- Generally the error_context is returned from _add_context_to_errors.
- The actual_error is useful for errors that are shared like invalid character.
- This can't filter out warnings like the other ones.

**pop_error_context**()

Remove the last scope from the error context.

### Notes

Modifies the error context of this reporter.

**push_error_context**(*context_type*, *context*)

Push a new error context to narrow down error scope.

**Parameters**

- **context_type** (ErrorContext) – A value from ErrorContext representing the type of scope.
- **context** (*str, int, or* HedString) – The main value for the context_type.

### Notes

The context depends on the context_type. For ErrorContext.FILE_NAME this would be the actual filename.

**reset_error_context**()

Reset all error context information to defaults.

### Notes

This function is mainly for testing and should not be needed with proper usage.

**val_error_unknown**(***kwargs*)

Default error handler if no error of this type was registered.

**Parameters**

- **args** (*args*) – List of non-keyword parameters (varies).
- **kwargs** (*kwargs*) – Keyword parameters (varies)

**Returns**
The error message.

**Return type**
str

### 3.1.3 hed.errors.error_types

#### Classes

| | |
|---|---|
| *ColumnErrors*() | |
| *DefinitionErrors*() | |
| *ErrorContext*() | Indicates the context this error took place in, each error potentially having multiple contexts |
| *ErrorSeverity*() | |
| *OnsetErrors*() | |
| *SchemaAttributeErrors*() | |
| *SchemaErrors*() | |
| *SchemaWarnings*() | |
| *SidecarErrors*() | |
| *ValidationErrors*() | |

#### 3.1.3.1 hed.errors.error_types.ColumnErrors

**class ColumnErrors**

Bases: object

**__init__**()

##### Methods

| | |
|---|---|
| *__init__*() | |

##### Attributes

| | |
|---|---|
| INVALID_COLUMN_REF | |
| MALFORMED_COLUMN_REF | |
| NESTED_COLUMN_REF | |
| SELF_COLUMN_REF | |

#### 3.1.3.2 hed.errors.error_types.DefinitionErrors

**class DefinitionErrors**

Bases: object

**__init__**()

##### Methods

| |
|---|
| *__init__*() |

##### Attributes

| |
|---|
| BAD_DEFINITION_LOCATION |
| BAD_PROP_IN_DEFINITION |
| DEF_TAG_IN_DEFINITION |
| DUPLICATE_DEFINITION |
| INVALID_DEFINITION_EXTENSION |
| NO_DEFINITION_CONTENTS |
| PLACEHOLDER_NO_TAKES_VALUE |
| WRONG_NUMBER_GROUPS |
| WRONG_NUMBER_PLACEHOLDER_TAGS |
| WRONG_NUMBER_TAGS |

#### 3.1.3.3 hed.errors.error_types.ErrorContext

**class ErrorContext**

Bases: object

Indicates the context this error took place in, each error potentially having multiple contexts

**__init__**()

#### Methods

| |
|---|
| *__init__*() |

#### Attributes

| |
|---|
| COLUMN |
| CUSTOM_TITLE |
| FILE_NAME |
| HED_STRING |
| LINE |
| ROW |
| SCHEMA_ATTRIBUTE |
| SCHEMA_SECTION |
| SCHEMA_TAG |
| SIDECAR_COLUMN_NAME |
| SIDECAR_KEY_NAME |

#### 3.1.3.4 hed.errors.error_types.ErrorSeverity

**class ErrorSeverity**

Bases: object

**__init__**()

#### Methods

| |
|---|
| *__init__*() |

#### Attributes

| |
|---|
| ERROR |
| WARNING |

#### 3.1.3.5 hed.errors.error_types.OnsetErrors

**class OnsetErrors**

Bases: object

**__init__**()

#### Methods

| |
|---|
| *__init__*() |

#### Attributes

| |
|---|
| INSET_BEFORE_ONSET |
| OFFSET_BEFORE_ONSET |
| ONSET_DEF_UNMATCHED |
| ONSET_NO_DEF_TAG_FOUND |
| ONSET_PLACEHOLDER_WRONG |
| ONSET_SAME_DEFS_ONE_ROW |
| ONSET_TAG_OUTSIDE_OF_GROUP |
| ONSET_TOO_MANY_DEFS |
| ONSET_WRONG_NUMBER_GROUPS |

#### 3.1.3.6 hed.errors.error_types.SchemaAttributeErrors

**class SchemaAttributeErrors**

Bases: object

**__init__**()

##### Methods

| |
|---|
| *__init__*() |

##### Attributes

| |
|---|
| SCHEMA_CHILD_OF_DEPRECATED |
| SCHEMA_DEFAULT_UNITS_INVALID |
| SCHEMA_DEPRECATED_INVALID |
| SCHEMA_RELATED_TAG_INVALID |
| SCHEMA_SUGGESTED_TAG_INVALID |
| SCHEMA_UNIT_CLASS_INVALID |
| SCHEMA_VALUE_CLASS_INVALID |

#### 3.1.3.7 hed.errors.error_types.SchemaErrors

**class SchemaErrors**

Bases: object

**__init__**()

##### Methods

| |
|---|
| *__init__*() |

#### Attributes

| |
|---|
| SCHEMA_ATTRIBUTE_INVALID |
| SCHEMA_DUPLICATE_FROM_LIBRARY |
| SCHEMA_DUPLICATE_NODE |

#### 3.1.3.8 hed.errors.error_types.SchemaWarnings

**class SchemaWarnings**

Bases: object

**__init__**()

#### Methods

| |
|---|
| *__init__*() |

#### Attributes

| |
|---|
| SCHEMA_CHARACTER_INVALID |
| SCHEMA_INVALID_ATTRIBUTE |
| SCHEMA_INVALID_CAPITALIZATION |
| SCHEMA_INVALID_CHARACTERS_IN_DESC |
| SCHEMA_INVALID_CHARACTERS_IN_TAG |
| SCHEMA_NON_PLACEHOLDER_HAS_CLASS |

#### 3.1.3.9 hed.errors.error_types.SidecarErrors

**class SidecarErrors**

Bases: object

**__init__**()

#### Methods

| |
|---|
| *__init__*() |

#### Attributes

| |
|---|
| BLANK_HED_STRING |
| INVALID_POUND_SIGNS_CATEGORY |
| INVALID_POUND_SIGNS_VALUE |
| SIDECAR_BRACES_INVALID |
| SIDECAR_HED_USED |
| SIDECAR_HED_USED_COLUMN |
| SIDECAR_NA_USED |
| UNKNOWN_COLUMN_TYPE |
| WRONG_HED_DATA_TYPE |

#### 3.1.3.10 hed.errors.error_types.ValidationErrors

**class ValidationErrors**

Bases: object

**__init__**()

#### Methods

| |
|---|
| *__init__*() |

#### Attributes

| |
|---|
| CHARACTER_INVALID |
| COMMA_MISSING |
| DEFINITION_INVALID |

continues on next page

Table 2 – continued from previous page

| |
|---|
| DEF_EXPAND_INVALID |
| DEF_INVALID |
| DUPLICATE_COLUMN_BETWEEN_SOURCES |
| DUPLICATE_COLUMN_IN_LIST |
| HED_BLANK_COLUMN |
| HED_DEF_EXPAND_INVALID |
| HED_DEF_EXPAND_UNMATCHED |
| HED_DEF_EXPAND_VALUE_EXTRA |
| HED_DEF_EXPAND_VALUE_MISSING |
| HED_DEF_UNMATCHED |
| HED_DEF_VALUE_EXTRA |
| HED_DEF_VALUE_MISSING |
| HED_GROUP_EMPTY |
| HED_LIBRARY_UNMATCHED |
| HED_MISSING_REQUIRED_COLUMN |
| HED_MULTIPLE_TOP_TAGS |
| HED_TAG_GROUP_TAG |
| HED_TAG_REPEATED |
| HED_TAG_REPEATED_GROUP |
| HED_TOP_LEVEL_TAG |
| HED_UNKNOWN_COLUMN |
| INVALID_PARENT_NODE |
| INVALID_TAG_CHARACTER |
| NODE_NAME_EMPTY |
| NO_VALID_TAG_FOUND |

continues on next page

Table 2 – continued from previous page

| |
|---|
| ONSET_OFFSET_INSET_ERROR |
| PARENTHESES_MISMATCH |
| PLACEHOLDER_INVALID |
| REQUIRED_TAG_MISSING |
| SIDECAR_AND_OTHER_COLUMNS |
| SIDECAR_INVALID |
| SIDECAR_KEY_MISSING |
| STYLE_WARNING |
| TAG_EMPTY |
| TAG_EXPRESSION_REPEATED |
| TAG_EXTENDED |
| TAG_EXTENSION_INVALID |
| TAG_GROUP_ERROR |
| TAG_INVALID |
| TAG_NAMESPACE_PREFIX_INVALID |
| TAG_NOT_UNIQUE |
| TAG_REQUIRES_CHILD |
| TILDES_UNSUPPORTED |
| UNITS_INVALID |
| UNITS_MISSING |
| VALUE_INVALID |
| VERSION_DEPRECATED |

### 3.1.4 hed.errors.exceptions

#### Classes

| |
|---|
| *HedExceptions*() |

#### 3.1.4.1 hed.errors.exceptions.HedExceptions

**class HedExceptions**

Bases: object

**__init__**()

##### Methods

| |
|---|
| *__init__*() |

### Attributes

| |
|---|
| BAD_COLUMN_NAMES |
| BAD_HED_LIBRARY_NAME |
| BAD_PARAMETERS |
| BAD_WITH_STANDARD |
| BAD_WITH_STANDARD_VERSION |
| CANNOT_PARSE_JSON |
| CANNOT_PARSE_XML |
| FILE_NOT_FOUND |
| GENERIC_ERROR |
| HED_END_INVALID |
| HED_SCHEMA_HEADER_INVALID |
| HED_SCHEMA_NODE_NAME_INVALID |
| HED_SCHEMA_VERSION_INVALID |
| HED_WIKI_DELIMITERS_INVALID |
| INVALID_DATAFRAME |
| INVALID_EXTENSION |
| INVALID_FILE_FORMAT |
| INVALID_HED_FORMAT |
| INVALID_SECTION_SEPARATOR |
| IN_LIBRARY_IN_UNMERGED |
| ROOTED_TAG_DOES_NOT_EXIST |
| ROOTED_TAG_HAS_PARENT |
| ROOTED_TAG_INVALID |
| SCHEMA_DUPLICATE_PREFIX |
| SCHEMA_END_INVALID |
| SCHEMA_HEADER_MISSING |
| SCHEMA_LIBRARY_INVALID |
| SCHEMA_START_MISSING |

### Exceptions

| | |
|---|---|
| *HedFileError*(code, message, filename[, issues]) | Exception raised when a file cannot be parsed due to being malformed, file IO, etc. |

#### 3.1.4.2 hed.errors.exceptions.HedFileError

**exception HedFileError**(*code*, *message*, *filename*, *issues=None*)

Exception raised when a file cannot be parsed due to being malformed, file IO, etc.

## 3.1.5 hed.errors.known_error_codes

## 3.1.6 hed.errors.schema_error_messages

### Functions

| | |
|---|---|
| *schema_error_SCHEMA_CHILD_OF_DEPRECATED*(...) | |
| *schema_error_SCHEMA_DEFAULT_UNITS_INVALID*(...) | |
| *schema_error_SCHEMA_DEPRECATED_INVALID*(...) | |
| *schema_error_SCHEMA_INVALID_ATTRIBUTE*(...) | |
| *schema_error_SCHEMA_SUGGESTED_TAG_INVALID*(...) | |
| *schema_error_SCHEMA_UNIT_CLASS_INVALID*(tag, ...) | |
| *schema_error_SCHEMA_VALUE_CLASS_INVALID*(tag, ...) | |
| *schema_error_hed_duplicate_from_library*(tag, ...) | |
| *schema_error_hed_duplicate_node*(tag, ...) | |
| *schema_error_unknown_attribute*(...) | |
| *schema_warning_SCHEMA_INVALID_CAPITALIZATION*(...) | |
| *schema_warning_invalid_chars_desc*(...) | |
| *schema_warning_invalid_chars_tag*(tag_name, ...) | |
| *schema_warning_non_placeholder_class*(...) | |

#### 3.1.6.1 hed.errors.schema_error_messages.schema_error_SCHEMA_CHILD_OF_DEPRECATED

schema_error_SCHEMA_CHILD_OF_DEPRECATED(*deprecated_tag*, *non_deprecated_child*)

#### 3.1.6.2 hed.errors.schema_error_messages.schema_error_SCHEMA_DEFAULT_UNITS_INVALID

schema_error_SCHEMA_DEFAULT_UNITS_INVALID(*tag*, *bad_unit*, *valid_units*)

#### 3.1.6.3 hed.errors.schema_error_messages.schema_error_SCHEMA_DEPRECATED_INVALID

schema_error_SCHEMA_DEPRECATED_INVALID(*tag_name*, *invalid_deprecated_version*)

#### 3.1.6.4 hed.errors.schema_error_messages.schema_error_SCHEMA_INVALID_ATTRIBUTE

schema_error_SCHEMA_INVALID_ATTRIBUTE(*tag_name*, *invalid_attribute_name*)

#### 3.1.6.5 hed.errors.schema_error_messages.schema_error_SCHEMA_SUGGESTED_TAG_INVALID

schema_error_SCHEMA_SUGGESTED_TAG_INVALID(*suggestedTag*, *invalidSuggestedTag*, *attribute_name*)

#### 3.1.6.6 hed.errors.schema_error_messages.schema_error_SCHEMA_UNIT_CLASS_INVALID

schema_error_SCHEMA_UNIT_CLASS_INVALID(*tag*, *unit_class*, *attribute_name*)

#### 3.1.6.7 hed.errors.schema_error_messages.schema_error_SCHEMA_VALUE_CLASS_INVALID

schema_error_SCHEMA_VALUE_CLASS_INVALID(*tag*, *unit_class*, *attribute_name*)

#### 3.1.6.8 hed.errors.schema_error_messages.schema_error_hed_duplicate_from_library

schema_error_hed_duplicate_from_library(*tag*, *duplicate_tag_list*, *section*)

#### 3.1.6.9 hed.errors.schema_error_messages.schema_error_hed_duplicate_node

schema_error_hed_duplicate_node(*tag*, *duplicate_tag_list*, *section*)

#### 3.1.6.10 hed.errors.schema_error_messages.schema_error_unknown_attribute

schema_error_unknown_attribute(*attribute_name*, *source_tag*)

#### 3.1.6.11 hed.errors.schema_error_messages.schema_warning_SCHEMA_INVALID_CAPITALIZATION

schema_warning_SCHEMA_INVALID_CAPITALIZATION(*tag_name*, *problem_char*, *char_index*)

#### 3.1.6.12 hed.errors.schema_error_messages.schema_warning_invalid_chars_desc

schema_warning_invalid_chars_desc(*desc_string*, *tag_name*, *problem_char*, *char_index*)

#### 3.1.6.13 hed.errors.schema_error_messages.schema_warning_invalid_chars_tag

schema_warning_invalid_chars_tag(*tag_name*, *problem_char*, *char_index*)

#### 3.1.6.14 hed.errors.schema_error_messages.schema_warning_non_placeholder_class

schema_warning_non_placeholder_class(*tag_name*, *invalid_attribute_name*)

## 3.2 hed.models

Data structures for HED tag handling.

### Modules

| | |
|---|---|
| hed.models.base_input | |
| hed.models.column_mapper | |
| hed.models.column_metadata | |
| hed.models.def_expand_gather | |
| hed.models.definition_dict | |
| hed.models.definition_entry | |
| hed.models.df_util | |
| hed.models.expression_parser | |
| hed.models.hed_group | |
| hed.models.hed_string | This module is used to split tags in a HED string. |
| hed.models.hed_tag | |
| hed.models.indexed_df | |
| hed.models.model_constants | |
| hed.models.sidecar | |
| hed.models.spreadsheet_input | |
| hed.models.string_util | |
| hed.models.tabular_input | |
| hed.models.timeseries_input | |

## 3.2.1 hed.models.base_input

### Classes

| | |
|---|---|
| BaseInput(file[, file_type, worksheet_name, ...]) | Superclass representing a basic columnar file. |

#### 3.2.1.1 hed.models.base_input.BaseInput

**class BaseInput**(*file*, *file_type=None*, *worksheet_name=None*, *has_column_names=True*, *mapper=None*, *name=None*, *allow_blank_names=True*)

Bases: `object`

Superclass representing a basic columnar file.

**__init__**(*file*, *file_type=None*, *worksheet_name=None*, *has_column_names=True*, *mapper=None*, *name=None*, *allow_blank_names=True*)

Constructor for the BaseInput class.

**Parameters**

- **file** (*str or file-like or pandas dataframe*) – An xlsx/tsv file to open.
- **file_type** (*str or None*) – ".xlsx" (Excel), ".tsv" or ".txt" (tab-separated text). Derived from file if file is a filename. Ignored if pandas dataframe.
- **worksheet_name** (*str or None*) – Name of Excel workbook worksheet name to use. (Not applicable to tsv files.)
- **has_column_names** (*bool*) – True if file has column names. This value is ignored if you pass in a pandas dataframe.
- **mapper** (*ColumnMapper or None*) – Indicates which columns have HED tags. See SpreadsheetInput or TabularInput for examples of how to use built-in a ColumnMapper.
- **name** (*str or None*) – Optional field for how this file will report errors.
- **allow_blank_names** (*bool*) – If True, column names can be blank

**Raises**

*HedFileError* –

- file is blank
- An invalid dataframe was passed with size 0
- An invalid extension was provided
- A duplicate or empty column name appears
- Cannot open the indicated file
- The specified worksheet name does not exist
- If the sidecar file or tabular file had invalid format and could not be read.

### Methods

| | |
|---|---|
| *__init__*(file[, file_type, worksheet_name, ...]) | Constructor for the BaseInput class. |
| *assemble*([mapper, skip_curly_braces]) | Assembles the hed strings |
| *column_metadata*() | Get the metadata for each column |
| *combine_dataframe*(dataframe) | Combines all columns in the given dataframe into a single HED string series, |
| *convert_to_form*(hed_schema, tag_form) | Convert all tags in underlying dataframe to the specified form. |
| *convert_to_long*(hed_schema) | Convert all tags in underlying dataframe to long form. |
| *convert_to_short*(hed_schema) | Convert all tags in underlying dataframe to short form. |
| *expand_defs*(hed_schema, def_dict) | Shrinks any def-expand found in the underlying dataframe. |
| *get_column_refs*() | Returns a list of column refs for this file. |
| *get_def_dict*(hed_schema[, extra_def_dicts]) | Returns the definition dict for this file |
| *get_worksheet*([worksheet_name]) | Get the requested worksheet. |
| *reset_mapper*(new_mapper) | Set mapper to a different view of the file. |
| *set_cell*(row_number, column_number, ...[, ...]) | Replace the specified cell with transformed text. |
| *shrink_defs*(hed_schema) | Shrinks any def-expand found in the underlying dataframe. |
| *to_csv*([file]) | Write to file or return as a string. |
| *to_excel*(file) | Output to an Excel file. |
| *validate*(hed_schema[, extra_def_dicts, ...]) | Creates a SpreadsheetValidator and returns all issues with this fil |

### Attributes

| |
|---|
| COMMA_DELIMITER |
| EXCEL_EXTENSION |
| FILE_EXTENSION |
| FILE_INPUT |
| STRING_INPUT |
| TAB_DELIMITER |
| TEXT_EXTENSION |

| | |
|---|---|
| *columns* | Returns a list of the column names. |
| *dataframe* | The underlying dataframe. |
| *dataframe_a* | Return the assembled dataframe |
| *has_column_names* | True if dataframe has column names. |
| *loaded_workbook* | The underlying loaded workbooks. |
| *name* | Name of the data. |
| *onsets* | Returns the onset column if it exists |
| *series_a* | Return the assembled dataframe as a series |
| *series_filtered* | Return the assembled dataframe as a series, with rows that have the same onset combined |
| *worksheet_name* | The worksheet name. |

**assemble**(*mapper=None*, *skip_curly_braces=False*)

Assembles the hed strings

**Parameters**

- **mapper** (*ColumnMapper or None*) – Generally pass none here unless you want special behavior.
- **skip_curly_braces** (*bool*) – If True, don't plug in curly brace values into columns.

**Returns**
the assembled dataframe

**Return type**
Dataframe

**column_metadata**()

Get the metadata for each column

**Returns**
number/ColumnMeta pairs

**Return type**
dict

**property columns**

Returns a list of the column names.

Empty if no column names.

**Returns**
the column names

**Return type**
columns(list)

`static combine_dataframe`(*dataframe*)

**Combines all columns in the given dataframe into a single HED string series,**
skipping empty columns and columns with empty strings.

**Parameters**
**dataframe** (*Dataframe*) – The dataframe to combine

**Returns**
the assembled series

**Return type**
Series

`convert_to_form`(*hed_schema*, *tag_form*)

Convert all tags in underlying dataframe to the specified form.

**Parameters**

- **hed_schema** (HedSchema) – The schema to use to convert tags.
- **tag_form** (*str*) – HedTag property to convert tags to. Most cases should use convert_to_short or convert_to_long below.

`convert_to_long`(*hed_schema*)

Convert all tags in underlying dataframe to long form.

**Parameters**
**hed_schema** (HedSchema *or None*) – The schema to use to convert tags.

`convert_to_short`(*hed_schema*)

Convert all tags in underlying dataframe to short form.

**Parameters**
**hed_schema** (HedSchema) – The schema to use to convert tags.

`property dataframe`

The underlying dataframe.

`property dataframe_a`

**Return the assembled dataframe**
Probably a placeholder name.

**Returns**
the assembled dataframe

**Return type**
Dataframe

`expand_defs`(*hed_schema*, *def_dict*)

Shrinks any def-expand found in the underlying dataframe.

**Parameters**

- **hed_schema** (HedSchema *or None*) – The schema to use to identify defs
- **def_dict** (DefinitionDict) – The definitions to expand

**get_column_refs**()

Returns a list of column refs for this file.

Default implementation returns none.

**Returns**
A list of unique column refs found

**Return type**
column_refs(list)

**get_def_dict**(*hed_schema*, *extra_def_dicts=None*)

Returns the definition dict for this file

Note: Baseclass implementation returns just extra_def_dicts.

**Parameters**

- **hed_schema** (HedSchema) – used to identify tags to find definitions(if needed)
- **extra_def_dicts** (*list, DefinitionDict, or None*) – Extra dicts to add to the list.

**Returns**
A single definition dict representing all the data(and extra def dicts)

**Return type**
*DefinitionDict*

**get_worksheet**(*worksheet_name=None*)

Get the requested worksheet.

**Parameters**
**worksheet_name** (*str or None*) – The name of the requested worksheet by name or the first one if None.

**Returns**
The workbook request.

**Return type**
openpyxl.workbook.Workbook

### Notes

If None, returns the first worksheet.

**Raises**
**KeyError** –

- The specified worksheet name does not exist

**property has_column_names**

True if dataframe has column names.

**property loaded_workbook**

The underlying loaded workbooks.

**property name**

Name of the data.

**property onsets**

Returns the onset column if it exists

**reset_mapper**(*new_mapper*)

Set mapper to a different view of the file.

**Parameters**

**new_mapper** (ColumnMapper) – A column mapper to be associated with this base input.

**property series_a**

Return the assembled dataframe as a series

**Returns**

the assembled dataframe with columns merged

**Return type**

Series

**property series_filtered**

Return the assembled dataframe as a series, with rows that have the same onset combined

**Returns**

the assembled dataframe with columns merged, and the rows filtered together

**Return type**

Series

**set_cell**(*row_number*, *column_number*, *new_string_obj*, *tag_form='short_tag'*)

Replace the specified cell with transformed text.

**Parameters**

- **row_number** (*int*) – The row number of the spreadsheet to set.
- **column_number** (*int*) – The column number of the spreadsheet to set.
- **new_string_obj** (HedString) – Object with text to put in the given cell.
- **tag_form** (*str*) – Version of the tags (short_tag, long_tag, base_tag, etc)

### Notes

Any attribute of a HedTag that returns a string is a valid value of tag_form.

**Raises**

- **ValueError** –
  - There is not a loaded dataframe
- **KeyError** –
  - the indicated row/column does not exist
- **AttributeError** –
  - The indicated tag_form is not an attribute of HedTag

**shrink_defs**(*hed_schema*)

Shrinks any def-expand found in the underlying dataframe.

**Parameters**
**hed_schema** (HedSchema *or None*) – The schema to use to identify defs

**to_csv**(*file=None*)

Write to file or return as a string.

**Parameters**
**file** (*str, file-like, or None*) – Location to save this file. If None, return as string.

**Returns**
None if file is given or the contents as a str if file is None.

**Return type**
None or str

**Raises**
**OSError** –

- Cannot open the indicated file

**to_excel**(*file*)

Output to an Excel file.

**Parameters**
**file** (*str or file-like*) – Location to save this base input.

**Raises**

- **ValueError** –
  - if empty file object was passed
- **OSError** –
  - Cannot open the indicated file

**validate**(*hed_schema*, *extra_def_dicts=None*, *name=None*, *error_handler=None*)

Creates a SpreadsheetValidator and returns all issues with this fil

**Parameters**

- **hed_schema** (HedSchema) – The schema to use for validation
- **extra_def_dicts** (*list of DefDict or DefDict*) – all definitions to use for validation
- **name** (*str*) – The name to report errors from this file as
- **error_handler** (ErrorHandler) – Error context to use. Creates a new one if None

**Returns**
A list of issues for hed string

**Return type**
issues (list of dict)

**property worksheet_name**

The worksheet name.

### 3.2.2 hed.models.column_mapper

#### Classes

| | |
|---|---|
| *ColumnMapper*([sidecar, tag_columns, ...]) | Mapping of a base input file columns into HED tags. |

#### 3.2.2.1 hed.models.column_mapper.ColumnMapper

**class ColumnMapper**(*sidecar=None*, *tag_columns=None*, *column_prefix_dictionary=None*, *optional_tag_columns=None*, *warn_on_missing_column=False*)

Bases: `object`

Mapping of a base input file columns into HED tags.

##### Notes

- All column numbers are 0 based.

**__init__**(*sidecar=None*, *tag_columns=None*, *column_prefix_dictionary=None*, *optional_tag_columns=None*, *warn_on_missing_column=False*)

Constructor for ColumnMapper.

**Parameters**

- **sidecar** (*Sidecar*) – A sidecar to gather column data from.
- **tag_columns** – (list): A list of ints or strings containing the columns that contain the HED tags. Sidecar column definitions will take precedent if there is a conflict with tag_columns.
- **column_prefix_dictionary** (*dict*) – Dictionary with keys that are column numbers/names and values are HED tag prefixes to prepend to the tags in that column before processing.
- **optional_tag_columns** (*list*) – A list of ints or strings containing the columns that contain the HED tags. If the column is otherwise unspecified, convert this column type to HEDTags.
- **warn_on_missing_column** (*bool*) – If True, issue mapping warnings on column names that are missing from the sidecar.

##### Notes

- All column numbers are 0 based.
- **The column_prefix_dictionary may be deprecated/renamed in the future.**
  - These are no longer prefixes, but rather converted to value columns: {"key": "Description", 1: "Label/"} will turn into value columns as {"key": "Description/#", 1: "Label/#"} It will be a validation issue if column 1 is called "key" in the above example. This means it no longer accepts anything but the value portion only in the columns.

### Methods

| | |
|---|---|
| *__init__*([sidecar, tag_columns, ...]) | Constructor for ColumnMapper. |
| *check_for_blank_names*(column_map, ...) | Validate there are no blank column names |
| *check_for_mapping_issues*([allow_blank_names]) | Find all issues given the current column_map, tag_columns, etc. |
| *get_column_mapping_issues*() | Get all the issues with finalizing column mapping(duplicate columns, missing required, etc) |
| *get_def_dict*(hed_schema[, extra_def_dicts]) | Return def dicts from every column description. |
| *get_tag_columns*() | Returns the column numbers or names that are mapped to be HedTags |
| *get_transformers*() | Return the transformers to use on a dataframe |
| *set_column_map*([new_column_map]) | Set the column number to name mapping. |
| *set_column_prefix_dictionary*(...[, ...]) | Sets the column prefix dictionary |
| *set_tag_columns*([tag_columns, ...]) | Set tag columns and optional tag columns |

### Attributes

| | |
|---|---|
| *column_prefix_dictionary* | Returns the column_prefix_dictionary with numbers turned into names where possible |
| *sidecar_column_data* | Pass through to get the sidecar ColumnMetadata |
| *tag_columns* | Returns the known tag and optional tag columns with numbers as names when possible |

**static check_for_blank_names**(*column_map*, *allow_blank_names*)

Validate there are no blank column names

**Parameters**

- **column_map** (*iterable*) – A list of column names
- **allow_blank_names** (*bool*) – Only find issues if this is true

**Returns**

A list of dicts, one per issue.

**Return type**

issues(list)

**check_for_mapping_issues**(*allow_blank_names=False*)

Find all issues given the current column_map, tag_columns, etc.

**Parameters**

**allow_blank_names** (*bool*) – Only flag blank names if False

**Returns**

Returns all issues found as a list of dicts

**Return type**

issue_list(list of dict)

**property column_prefix_dictionary**

Returns the column_prefix_dictionary with numbers turned into names where possible

**Returns**

A column_prefix_dictionary with column labels as keys

**Return type**
column_prefix_dictionary(list of str or int)

`get_column_mapping_issues()`

Get all the issues with finalizing column mapping(duplicate columns, missing required, etc)

### Notes

- This is deprecated and now a wrapper for "check_for_mapping_issues()"

**Returns**
A list dictionaries of all issues found from mapping column names to numbers.

**Return type**
list

`get_def_dict(hed_schema, extra_def_dicts=None)`

Return def dicts from every column description.

**Parameters**

- **hed_schema** (*Schema*) – A HED schema object to use for extracting definitions.
- **extra_def_dicts** (*list, DefinitionDict, or None*) – Extra dicts to add to the list.

**Returns**
A single definition dict representing all the data(and extra def dicts)

**Return type**
*DefinitionDict*

`get_tag_columns()`

Returns the column numbers or names that are mapped to be HedTags

Note: This is NOT the tag_columns or optional_tag_columns parameter, though they set it.

**Returns**

**A list of column numbers or names that are ColumnType.HedTags.**
0-based if integer-based, otherwise column name.

**Return type**
column_identifiers(list)

`get_transformers()`

Return the transformers to use on a dataframe

**Returns**
dict({str or int: func}): the functions to use to transform each column need_categorical(list of int): a list of columns to treat as categoriacl

**Return type**
tuple(dict, list)

`set_column_map(new_column_map=None)`

Set the column number to name mapping.

**Parameters**
**new_column_map** (*list or dict*) – Either an ordered list of the column names or column_number:column name dictionary. In both cases, column numbers start at 0

**Returns**
List of issues. Each issue is a dictionary.

**Return type**
list

**set_column_prefix_dictionary**(*column_prefix_dictionary*, *finalize_mapping=True*)

Sets the column prefix dictionary

**set_tag_columns**(*tag_columns=None*, *optional_tag_columns=None*, *finalize_mapping=True*)

Set tag columns and optional tag columns

**Parameters**

- **tag_columns** (*list*) – A list of ints or strings containing the columns that contain the HED tags. If None, clears existing tag_columns
- **optional_tag_columns** (*list*) – A list of ints or strings containing the columns that contain the HED tags, but not an error if missing. If None, clears existing tag_columns
- **finalize_mapping** (*bool*) – Re-generate the internal mapping if True, otherwise no effect until finalize.

**property sidecar_column_data**

Pass through to get the sidecar ColumnMetadata

**Returns**
ColumnMetadata}): the column metadata defined by this sidecar

**Return type**
dict({str

**property tag_columns**

Returns the known tag and optional tag columns with numbers as names when possible

**Returns**
A list of all tag and optional tag columns as labels

**Return type**
tag_columns(list of str or int)

### 3.2.3 hed.models.column_metadata

#### Classes

| | |
|---|---|
| *ColumnMetadata*([column_type, name, source]) | Column in a ColumnMapper. |
| *ColumnType*(value) | The overall column_type of a column in column mapper, e.g. |

#### 3.2.3.1 hed.models.column_metadata.ColumnMetadata

**class ColumnMetadata**(*column_type=None*, *name=None*, *source=None*)

Bases: `object`

Column in a ColumnMapper.

**__init__**(*column_type=None*, *name=None*, *source=None*)

A single column entry in the column mapper.

**Parameters**

- **column_type** (*ColumnType or None*) – How to treat this column when reading data.
- **name** (*str, int, or None*) – The column_name or column number identifying this column. If name is a string, you'll need to use a column map to set the number later.
- **source** (*dict or str or None*) – Either the entire loaded json sidecar or a single HED string

### Methods

| | |
|---|---|
| *__init__*([column_type, name, source]) | A single column entry in the column mapper. |
| *expected_pound_sign_count*(column_type) | Return how many pound signs a column string should have. |
| *get_hed_strings*() | Returns the hed strings for this entry as a series. |
| *set_hed_strings*(new_strings) | Sets the hed strings for this entry. |

### Attributes

| | |
|---|---|
| *hed_dict* | The hed strings for any given entry. |
| *source_dict* | The raw dict for this entry(if it exists) |

**static expected_pound_sign_count**(*column_type*)

Return how many pound signs a column string should have.

**Parameters**
**column_type** (*ColumnType*) – The type of the column

**Returns**
expected_count(int): The expected count. 0 or 1 error_type(str): The type of the error we should issue

**Return type**
tuple

**get_hed_strings**()

Returns the hed strings for this entry as a series.

**Returns**
the hed strings for this series.(potentially empty)

**Return type**
hed_strings(pd.Series)

property hed_dict

The hed strings for any given entry.

**Returns**
A string or dict of strings for this column

**Return type**
dict or str

set_hed_strings(*new_strings*)

Sets the hed strings for this entry.

**Parameters**
**new_strings** (*pd.Series, dict, or str*) – The hed strings to set. This should generally be the return value from get_hed_strings

**Returns**
the hed strings for this series.(potentially empty)

**Return type**
hed_strings(pd.Series)

property source_dict

The raw dict for this entry(if it exists)

**Returns**
A string or dict of strings for this column

**Return type**
dict or str

#### 3.2.3.2 hed.models.column_metadata.ColumnType

class ColumnType(*value*)

Bases: Enum

The overall column_type of a column in column mapper, e.g. treat it as HED tags.

Mostly internal to column mapper related code

__init__()

##### Attributes

| |
|---|
| Unknown |
| Ignore |
| Categorical |
| Value |
| HEDTags |

### 3.2.4 hed.models.def_expand_gather

#### Classes

| | |
|---|---|
| *AmbiguousDef*() | |
| *DefExpandGatherer*(hed_schema[, known_defs, ...]) | Class for gathering definitions from a series of def-expands, including possibly ambiguous ones |

#### 3.2.4.1 hed.models.def_expand_gather.AmbiguousDef

**class AmbiguousDef**

Bases: object

**__init__**()

##### Methods

| | |
|---|---|
| *__init__*() | |
| add_def(def_tag, def_expand_group) | |
| get_group() | |
| *validate*() | Validate the given ambiguous definition |

**validate**()

Validate the given ambiguous definition

**Returns**
True if this is a valid definition with exactly 1 placeholder.

**Return type**
bool

**Raises**
**ValueError** – Raised if this is an invalid(not ambiguous) definition.

#### 3.2.4.2 hed.models.def_expand_gather.DefExpandGatherer

**class DefExpandGatherer**(*hed_schema*, *known_defs=None*, *ambiguous_defs=None*, *errors=None*)

Bases: object

Class for gathering definitions from a series of def-expands, including possibly ambiguous ones

**__init__**(*hed_schema*, *known_defs=None*, *ambiguous_defs=None*, *errors=None*)

Initialize the DefExpandGatherer class.

**Parameters**

- **hed_schema** (HedSchema) – The HED schema to be used for processing.
- **known_defs** (*dict, optional*) – A dictionary of known definitions.

- **ambiguous_defs** (*dict, optional*) – A dictionary of ambiguous def-expand definitions.

#### Methods

| | |
|---|---|
| *__init__*(hed_schema[, known_defs, ...]) | Initialize the DefExpandGatherer class. |
| *get_ambiguous_group*(ambiguous_def) | Turns an entry in the ambiguous_defs dict into a single HedGroup |
| *process_def_expands*(hed_strings[, known_defs]) | Process the HED strings containing def-expand tags. |

**static get_ambiguous_group**(*ambiguous_def*)

Turns an entry in the ambiguous_defs dict into a single HedGroup

**Returns**
the ambiguous definition with known placeholders filled in

**Return type**
*HedGroup*

**process_def_expands**(*hed_strings*, *known_defs=None*)

Process the HED strings containing def-expand tags.

**Parameters**

- **hed_strings** (*pd.Series or list*) – A Pandas Series or list of HED strings to be processed.
- **known_defs** (*dict, optional*) – A dictionary of known definitions to be added.

**Returns**
A tuple containing the DefinitionDict, ambiguous definitions, and errors.

**Return type**
tuple

### 3.2.5 hed.models.definition_dict

#### Classes

| | |
|---|---|
| *DefinitionDict*([def_dicts, hed_schema]) | Gathers definitions from a single source. |

#### 3.2.5.1 hed.models.definition_dict.DefinitionDict

**class DefinitionDict**(*def_dicts=None*, *hed_schema=None*)

Bases: object

Gathers definitions from a single source.

**__init__**(*def_dicts=None*, *hed_schema=None*)

Definitions to be considered a single source.

**Parameters**

- **def_dicts** (*str or list or* DefinitionDict) – DefDict or list of DefDicts/strings or a single string whose definitions should be added.
- **hed_schema** (HedSchema *or None*) – Required if passing strings or lists of strings, unused otherwise.

**Raises**

**TypeError** –

- Bad type passed as def_dicts

### Methods

| | |
|---|---|
| *__init__*([def_dicts, hed_schema]) | Definitions to be considered a single source. |
| *add_definitions*(def_dicts[, hed_schema]) | Add definitions from dict(s) to this dict. |
| *check_for_definitions*(hed_string_obj[, ...]) | Check string for definition tags, adding them to self. |
| *construct_def_tag*(hed_tag) | Identify def/def-expand tag contents in the given HedTag. |
| *construct_def_tags*(hed_string_obj) | Identify def/def-expand tag contents in the given string. |
| *get*(def_name) | Get the definition entry for the definition name. |
| *get_as_strings*(def_dict) | Convert the entries to strings of the contents |
| *items*() | Returns the dictionary of definitions |

### Attributes

| | |
|---|---|
| *issues* | Returns issues about duplicate definitions. |

**add_definitions**(*def_dicts*, *hed_schema=None*)

Add definitions from dict(s) to this dict.

**Parameters**

- **def_dicts** (*list, DefinitionDict, or dict*) –

  **DefinitionDict or list of DefinitionDicts/strings/dicts whose definitions should be added.**

  Note dict form expects DefinitionEntries in the same form as a DefinitionDict

- **hed_schema** (HedSchema *or None*) – Required if passing strings or lists of strings, unused otherwise.

**Raises**

**TypeError** –

- Bad type passed as def_dicts

**check_for_definitions**(*hed_string_obj*, *error_handler=None*)

Check string for definition tags, adding them to self.

**Parameters**

- **hed_string_obj** (HedString) – A single hed string to gather definitions from.
- **error_handler** (ErrorHandler *or None*) – Error context used to identify where definitions are found.

**Returns**
List of issues encountered in checking for definitions. Each issue is a dictionary.

**Return type**
list

`construct_def_tag`(*hed_tag*)

Identify def/def-expand tag contents in the given HedTag.

**Parameters**
**hed_tag** (HedTag) – The hed tag to identify definition contents in

`construct_def_tags`(*hed_string_obj*)

Identify def/def-expand tag contents in the given string.

**Parameters**
**hed_string_obj** (HedString) – The hed string to identify definition contents in

`get`(*def_name*)

Get the definition entry for the definition name.

Not case-sensitive

**Parameters**
**def_name** (*str*) – Name of the definition to retrieve.

**Returns**
Definition entry for the requested definition.

**Return type**
*DefinitionEntry*

`static get_as_strings`(*def_dict*)

Convert the entries to strings of the contents

**Parameters**
**def_dict** (DefinitionDict *or dict*) – A dict of definitions

**Returns**
str): definition name and contents

**Return type**
dict(str

`property issues`

Returns issues about duplicate definitions.

`items`()

Returns the dictionary of definitions

Alias for .defs.items()

**Returns**
DefinitionEntry}): A list of definitions

**Return type**
def_entries({str

## 3.2.6 hed.models.definition_entry

### Classes

| | |
|---|---|
| *DefinitionEntry*(name, contents, takes_value, ...) | A single definition. |

#### 3.2.6.1 hed.models.definition_entry.DefinitionEntry

**class DefinitionEntry**(*name*, *contents*, *takes_value*, *source_context*)

Bases: `object`

A single definition.

**__init__**(*name*, *contents*, *takes_value*, *source_context*)

Initialize info for a single definition.

**Parameters**

- **name** (*str*) – The label portion of this name (not including Definition/).
- **contents** (HedGroup) – The contents of this definition.
- **takes_value** (*bool*) – If True, expects ONE tag to have a single # sign in it.
- **source_context** (*list, None*) – List (stack) of dictionaries giving context for reporting errors.

### Methods

| | |
|---|---|
| *__init__*(name, contents, takes_value, ...) | Initialize info for a single definition. |
| *get_definition*(replace_tag[, ...]) | Return a copy of the definition with the tag expanded and the placeholder plugged in. |

**get_definition**(*replace_tag*, *placeholder_value=None*, *return_copy_of_tag=False*)

Return a copy of the definition with the tag expanded and the placeholder plugged in.

Returns None if placeholder_value passed when it doesn't take value, or vice versa.

**Parameters**

- **replace_tag** (HedTag) – The def hed tag to replace with an expanded version
- **placeholder_value** (*str or None*) – If present and required, will replace any pound signs in the definition contents.
- **return_copy_of_tag** (*bool*) – Set to true for validation

**Returns**

The contents of this definition(including the def tag itself)

**Return type**

*HedGroup*

**Raises**

**ValueError** –

- Something internally went wrong with finding the placeholder tag. This should not be possible.

## 3.2.7 hed.models.df_util

### Functions

| | |
|---|---|
| *convert_to_form*(df, hed_schema, tag_form[, ...]) | Convert all tags in underlying dataframe to the specified form (in place). |
| *expand_defs*(df, hed_schema, def_dict[, columns]) | Expands any def tags found in the dataframe. |
| *get_assembled*(tabular_file, sidecar, hed_schema) | Load a tabular file and its associated HED sidecar file. |
| *process_def_expands*(hed_strings, hed_schema) | Gather def-expand tags in the strings/compare with known definitions to find any differences |
| *shrink_defs*(df, hed_schema[, columns]) | Shrink (in place) any def-expand tags found in the specified columns in the dataframe. |

#### 3.2.7.1 hed.models.df_util.convert_to_form

**convert_to_form**(*df*, *hed_schema*, *tag_form*, *columns=None*)

Convert all tags in underlying dataframe to the specified form (in place).

**Parameters**

- **df** (*pd.Dataframe or pd.Series*) – The dataframe or series to modify
- **hed_schema** (HedSchema) – The schema to use to convert tags.
- **tag_form** (*str*) – HedTag property to convert tags to.
- **columns** (*list*) – The columns to modify on the dataframe.

#### 3.2.7.2 hed.models.df_util.expand_defs

**expand_defs**(*df*, *hed_schema*, *def_dict*, *columns=None*)

Expands any def tags found in the dataframe.

Converts in place

**Parameters**

- **df** (*pd.Dataframe or pd.Series*) – The dataframe or series to modify
- **hed_schema** (HedSchema *or None*) – The schema to use to identify defs
- **def_dict** (DefinitionDict) – The definitions to expand
- **columns** (*list or None*) – The columns to modify on the dataframe

#### 3.2.7.3 hed.models.df_util.get_assembled

**get_assembled**(*tabular_file*, *sidecar*, *hed_schema*, *extra_def_dicts=None*, *join_columns=True*, *shrink_defs=False*, *expand_defs=True*)

Load a tabular file and its associated HED sidecar file.

**Parameters**

- **tabular_file** – str or TabularInput The path to the tabular file, or a TabularInput object representing it.
- **sidecar** – str or Sidecar The path to the sidecar file, or a Sidecar object representing it.
- **hed_schema** – HedSchema If str, will attempt to load as a version if it doesn't have a valid extension.
- **extra_def_dicts** – list of DefinitionDict, optional Any extra DefinitionDict objects to use when parsing the HED tags.
- **join_columns** – bool If true, join all HED columns into one.
- **shrink_defs** – bool Shrink any def-expand tags found
- **expand_defs** – bool Expand any def tags found

**Returns**

hed_strings(list of HedStrings):A list of HedStrings or a list of lists of HedStrings def_dict(DefinitionDict): The definitions from this Sidecar

**Return type**

tuple

#### 3.2.7.4 hed.models.df_util.process_def_expands

**process_def_expands**(*hed_strings*, *hed_schema*, *known_defs=None*, *ambiguous_defs=None*)

Gather def-expand tags in the strings/compare with known definitions to find any differences

**Parameters**

- **hed_strings** (*list or pd.Series*) – A list of HED strings to process.
- **hed_schema** (HedSchema) – The schema to use
- **known_defs** (DefinitionDict *or list or str or None*) – A DefinitionDict or anything its constructor takes. These are the known definitions going in, that must match perfectly.
- **ambiguous_defs** (*dict*) – A dictionary containing ambiguous definitions format TBD. Currently def name key: list of lists of HED tags values

**Returns**

A tuple containing the DefinitionDict, ambiguous definitions, and errors.

**Return type**

tuple

#### 3.2.7.5 hed.models.df_util.shrink_defs

**shrink_defs**(*df*, *hed_schema*, *columns=None*)

Shrink (in place) any def-expand tags found in the specified columns in the dataframe.

**Parameters**

- **df** (*pd.Dataframe or pd.Series*) – The dataframe or series to modify
- **hed_schema** (*HedSchema or None*) – The schema to use to identify defs.
- **columns** (*list or None*) – The columns to modify on the dataframe.

### 3.2.8 hed.models.expression_parser

#### Classes

| | |
|---|---|
| *Expression*(token[, left, right]) | |
| *ExpressionAnd*(token[, left, right]) | |
| *ExpressionContainingGroup*(token[, left, right]) | |
| *ExpressionDescendantGroup*(token[, left, right]) | |
| *ExpressionExactMatch*(token[, left, right]) | |
| *ExpressionNegation*(token[, left, right]) | |
| *ExpressionOr*(token[, left, right]) | |
| *ExpressionWildcardNew*(token[, left, right]) | |
| *QueryParser*(expression_string) | Parse a search expression into a form than can be used to search a hed string. |
| *Token*(text) | |
| *search_result*(group, tag) | |

#### 3.2.8.1 hed.models.expression_parser.Expression

**class Expression**(*token*, *left=None*, *right=None*)

Bases: `object`

**__init__**(*token*, *left=None*, *right=None*)

#### Methods

| |
|---|
| *__init__*(token[, left, right]) |
| handle_expr(hed_group[, exact]) |

#### 3.2.8.2 hed.models.expression_parser.ExpressionAnd

**class ExpressionAnd**(*token*, *left=None*, *right=None*)

Bases: *Expression*

**__init__**(*token*, *left=None*, *right=None*)

#### Methods

| |
|---|
| *__init__*(token[, left, right]) |
| handle_expr(hed_group[, exact]) |
| merge_groups(groups1, groups2) |

#### 3.2.8.3 hed.models.expression_parser.ExpressionContainingGroup

**class ExpressionContainingGroup**(*token*, *left=None*, *right=None*)

Bases: *Expression*

**__init__**(*token*, *left=None*, *right=None*)

#### Methods

| |
|---|
| *__init__*(token[, left, right]) |
| handle_expr(hed_group[, exact]) |

#### 3.2.8.4 hed.models.expression_parser.ExpressionDescendantGroup

**class ExpressionDescendantGroup**(*token*, *left=None*, *right=None*)

Bases: *Expression*

**__init__**(*token*, *left=None*, *right=None*)

#### Methods

| |
|---|
| *__init__*(token[, left, right]) |
| handle_expr(hed_group[, exact]) |

#### 3.2.8.5 hed.models.expression_parser.ExpressionExactMatch

**class ExpressionExactMatch**(*token*, *left=None*, *right=None*)

Bases: *Expression*

**__init__**(*token*, *left=None*, *right=None*)

#### Methods

| |
|---|
| *__init__*(token[, left, right]) |
| handle_expr(hed_group[, exact]) |

#### 3.2.8.6 hed.models.expression_parser.ExpressionNegation

**class ExpressionNegation**(*token*, *left=None*, *right=None*)

Bases: *Expression*

**__init__**(*token*, *left=None*, *right=None*)

#### Methods

| |
|---|
| *__init__*(token[, left, right]) |
| handle_expr(hed_group[, exact]) |

#### 3.2.8.7 hed.models.expression_parser.ExpressionOr

**class ExpressionOr**(*token*, *left=None*, *right=None*)

Bases: *Expression*

**__init__**(*token*, *left=None*, *right=None*)

#### Methods

| |
|---|
| *__init__*(token[, left, right]) |
| handle_expr(hed_group[, exact]) |

#### 3.2.8.8 hed.models.expression_parser.ExpressionWildcardNew

**class ExpressionWildcardNew**(*token*, *left=None*, *right=None*)

Bases: *Expression*

**__init__**(*token*, *left=None*, *right=None*)

#### Methods

| |
|---|
| *__init__*(token[, left, right]) |
| handle_expr(hed_group[, exact]) |

#### 3.2.8.9 hed.models.expression_parser.QueryParser

**class QueryParser**(*expression_string*)

Bases: object

Parse a search expression into a form than can be used to search a hed string.

**__init__**(*expression_string*)

Compiles a QueryParser for a particular expression, so it can be used to search hed strings.

Basic Input Examples:

'Event' - Finds any strings with Event, or a descendent tag of Event such as Sensory-event

'Event and Action' - Find any strings with Event and Action, including descendant tags

'Event or Action' - Same as above, but it has either

'"Event"' - Finds the Event tag, but not any descendent tags

'Def/DefName/*' - Find Def/DefName instances with placeholders, regardless of the value of the placeholder

'Eve*' - Find any short tags that begin with Eve*, such as Event, but not Sensory-event

'[Event and Action]' - Find a group that contains both Event and Action(at any level)

'[[Event and Action]]' - Find a group with Event And Action at the same level.

Practical Complex Example:

**[[{(Onset or Offset), (Def or [[Def-expand]]): ???}]] - A group with an onset tag,**
a def tag or def-expand group, and an optional wildcard group

**Parameters**
**expression_string** (*str*) – The query string

#### Methods

| | |
|---|---|
| *__init__*(expression_string) | Compiles a QueryParser for a particular expression, so it can be used to search hed strings. |
| current_token() | |
| search(hed_string_obj) | |

#### 3.2.8.10 hed.models.expression_parser.Token

**class Token**(*text*)

Bases: object

**__init__**(*text*)

#### Methods

| | |
|---|---|
| *__init__*(text) | |

#### Attributes

| |
|---|
| And |
| ContainingGroup |
| ContainingGroupEnd |
| DescendantGroup |
| DescendantGroupEnd |
| ExactMatch |
| ExactMatchEnd |
| ExactMatchOptional |
| LogicalGroup |
| LogicalGroupEnd |
| LogicalNegation |
| NotInLine |
| Or |
| Tag |
| Wildcard |

#### 3.2.8.11 hed.models.expression_parser.search_result

**class search_result**(*group*, *tag*)

Bases: object

**__init__**(*group*, *tag*)

#### Methods

| |
|---|
| *__init__*(group, tag) |
| get_groups_only() |
| get_tags_only() |
| has_same_tags(other) |
| merge_result(other) |

### 3.2.9 hed.models.hed_group

#### Classes

| | |
|---|---|
| *HedGroup*([hed_string, startpos, endpos, ...]) | A single parenthesized hed string. |

#### 3.2.9.1 hed.models.hed_group.HedGroup

**class HedGroup**(*hed_string=''*, *startpos=None*, *endpos=None*, *contents=None*)

Bases: object

A single parenthesized hed string.

**__init__**(*hed_string=''*, *startpos=None*, *endpos=None*, *contents=None*)

Return an empty HedGroup object.

**Parameters**

- **hed_string** (*str or None*) – Source hed string for this group.
- **startpos** (*int or None*) – Starting index of group(including parentheses) in hed_string.
- **endpos** (*int or None*) – Position after the end (including parentheses) in hed_string.
- **contents** (*list or None*) – A list of HedTags and/or HedGroups that will be set as the contents of this group. Mostly used during definition expansion.

### Methods

| | |
|---|---|
| *__init__*([hed_string, startpos, endpos, ...]) | Return an empty HedGroup object. |
| *append*(tag_or_group) | Add a tag or group to this group. |
| *check_if_in_original*(tag_or_group) | Check if the tag or group in original string. |
| *copy*() | Return a deep copy of this group. |
| *find_def_tags*([recursive, include_groups]) | Find def and def-expand tags |
| *find_exact_tags*(exact_tags[, recursive, ...]) | Find the given tags. |
| *find_placeholder_tag*() | Return a placeholder tag, if present in this group. |
| *find_tags*(search_tags[, recursive, ...]) | Find the base tags and their containing groups. |
| *find_tags_with_term*(term[, recursive, ...]) | Find any tags that contain the given term. |
| *find_wildcard_tags*(search_tags[, recursive, ...]) | Find the tags and their containing groups. |
| *get_all_groups*([also_return_depth]) | Return HedGroups, including descendants and self. |
| *get_all_tags*() | Return HedTags, including descendants. |
| *get_as_form*(tag_attribute) | Get the string corresponding to the specified form. |
| *get_as_long*() | Return this HedGroup as a long tag string. |
| *get_as_short*() | Return this HedGroup as a short tag string. |
| *get_first_group*() | Returns the first group in this hed string or group. |
| *get_original_hed_string*() | Get the original hed string. |
| *groups*() | Return the direct child groups of this group. |
| *lower*() | Convenience function, equivalent to str(self).lower() |
| *remove*(items_to_remove) | Remove any tags/groups in items_to_remove. |
| *replace*(item_to_replace, new_contents) | Replace an existing tag or group. |
| *sort*() | Sort the tags and groups in this HedString in a consistent order. |
| *sorted*() | Returns a sorted copy of this hed group |
| *tags*() | Return the direct child tags of this group. |

### Attributes

| | |
|---|---|
| *is_group* | True if this is a parenthesized group. |
| *span* | Return the source span. |

**append**(*tag_or_group*)

Add a tag or group to this group.

**Parameters**
**tag_or_group** (HedTag *or* HedGroup) – The new object to add to this group.

**check_if_in_original**(*tag_or_group*)

Check if the tag or group in original string.

**Parameters**
**tag_or_group** (HedTag *or* HedGroup) – The HedTag or HedGroup to be looked for in this group.

**Returns**
True if in this group.

**Return type**
bool

**copy**()

Return a deep copy of this group.

**Returns**
The copied group.

**Return type**
*HedGroup*

**find_def_tags**(*recursive=False*, *include_groups=3*)

Find def and def-expand tags

**Parameters**

- **recursive** (*bool*) – If true, also check subgroups.
- **include_groups** (*int, 0, 1, 2, 3*) – options for return values If 0: Return only def and def expand tags/. If 1: Return only def tags and def-expand groups. If 2: Return only groups containing defs, or def-expand groups. If 3 or any other value: Return all 3 as a tuple.

**Returns**
A list of tuples. The contents depend on the values of the include_group.

**Return type**
list

**find_exact_tags**(*exact_tags*, *recursive=False*, *include_groups=1*)

Find the given tags. This will only find complete matches, any extension or value must also match.

**Parameters**

- **exact_tags** (*list of* HedTag) – A container of tags to locate.
- **recursive** (*bool*) – If true, also check subgroups.
- **include_groups** (*bool*) – 0, 1 or 2 If 0: Return only tags If 1: Return only groups If 2 or any other value: Return both

**Returns**
A list of tuples. The contents depend on the values of the include_group.

**Return type**
list

**find_placeholder_tag**()

Return a placeholder tag, if present in this group.

**Returns**
The placeholder tag if found.

**Return type**
*HedTag* or None

### Notes

- Assumes a valid HedString with no erroneous "#" characters.

**find_tags**(*search_tags*, *recursive=False*, *include_groups=2*)

Find the base tags and their containing groups. This searches by short_base_tag, ignoring any ancestors or extensions/values.

**Parameters**

- **search_tags** (*container*) – A container of short_base_tags to locate
- **recursive** (*bool*) – If true, also check subgroups.
- **include_groups** (*0, 1 or 2*) – Specify return values. If 0: return a list of the HedTags. If 1: return a list of the HedGroups containing the HedTags. If 2: return a list of tuples (HedTag, HedGroup) for the found tags.

**Returns**
The contents of the list depends on the value of include_groups.

**Return type**
list

**find_tags_with_term**(*term*, *recursive=False*, *include_groups=2*)

Find any tags that contain the given term.

Note: This can only find identified tags.

**Parameters**

- **term** (*str*) – A single term to search for.
- **recursive** (*bool*) – If true, recursively check subgroups.
- **include_groups** (*0, 1 or 2*) – Controls return values If 0: Return only tags If 1: Return only groups If 2 or any other value: Return both

**Return type**
list

**find_wildcard_tags**(*search_tags*, *recursive=False*, *include_groups=2*)

Find the tags and their containing groups.

This searches tag.short_tag, with an implicit wildcard on the end.

e.g. "Eve" will find Event, but not Sensory-event

**Parameters**

- **search_tags** (*container*) – A container of the starts of short tags to search.
- **recursive** (*bool*) – If true, also check subgroups.
- **include_groups** (*0, 1 or 2*) – Specify return values. If 0: return a list of the HedTags. If 1: return a list of the HedGroups containing the HedTags. If 2: return a list of tuples (HedTag, HedGroup) for the found tags.

**Returns**
The contents of the list depends on the value of include_groups.

**Return type**
list

get_all_groups(*also_return_depth=False*)

Return HedGroups, including descendants and self.

**Parameters**
**also_return_depth** (*bool*) – If True, yield tuples (group, depth) rather than just groups.

**Returns**
The list of all HedGroups in this group, including descendants and self.

**Return type**
list

get_all_tags()

Return HedTags, including descendants.

**Returns**
A list of all the tags in this group including descendants.

**Return type**
list

get_as_form(*tag_attribute*)

Get the string corresponding to the specified form.

**Parameters**
**tag_attribute** (*str*) – The hed_tag property to use to construct the string (usually short_tag or long_tag).

**Returns**
The constructed string after transformation

**Return type**
str

get_as_long()

Return this HedGroup as a long tag string.

**Returns**
The group as a string with all tags as long tags.

**Return type**
str

get_as_short()

Return this HedGroup as a short tag string.

**Returns**
The group as a string with all tags as short tags.

**Return type**
str

get_first_group()

Returns the first group in this hed string or group.

Useful for things like Def-expand where they only have a single group.

Raises a ValueError if there are no groups.

**Returns**
The first group

**Return type**
*HedGroup*

**get_original_hed_string**()

Get the original hed string.

**Returns**
The original string with no modification.

**Return type**
str

**groups**()

Return the direct child groups of this group.

**Returns**
All groups directly in this group, filtering out HedTag children.

**Return type**
list

**property is_group**

True if this is a parenthesized group.

**lower**()

Convenience function, equivalent to str(self).lower()

**remove**(*items_to_remove: Iterable[Union[*HedTag, HedGroup*]]*)

Remove any tags/groups in items_to_remove.

**Parameters**
**items_to_remove** (*list*) – List of HedGroups and/or HedTags to remove by identity.

### Notes

- Any groups that become empty will also be pruned.
- If you pass a child and parent group, the child will also be removed from the parent.

**static replace**(*item_to_replace*, *new_contents*)

Replace an existing tag or group.

Note: This is a static method that relies on the parent attribute of item_to_replace.

**Parameters**

- **item_to_replace** (HedTag *or* HedGroup) – The item to replace must exist or this will raise an error.
- **new_contents** (HedTag *or* HedGroup) – Replacement contents.

**Raises**

- **KeyError** –
  - item_to_replace does not exist
- **AttributeError** –
  - item_to_replace has no parent set

**sort**()

Sort the tags and groups in this HedString in a consistent order.

**sorted**()

Returns a sorted copy of this hed group

**Returns**
The sorted copy

**Return type**
sorted_copy (*HedGroup*)

**property span**

Return the source span.

**Returns**
start index of the group (including parentheses) from the source string. int: end index of the group (including parentheses) from the source string.

**Return type**
int

**tags**()

Return the direct child tags of this group.

**Returns**
All tags directly in this group, filtering out HedGroup children.

**Return type**
list

### 3.2.10 hed.models.hed_string

This module is used to split tags in a HED string.

#### Classes

| | |
|---|---|
| *HedString*(hed_string, hed_schema[, ...]) | A HED string. |

#### 3.2.10.1 hed.models.hed_string.HedString

**class HedString**(*hed_string*, *hed_schema*, *def_dict=None*, *_contents=None*)

Bases: *HedGroup*

A HED string.

**__init__**(*hed_string*, *hed_schema*, *def_dict=None*, *_contents=None*)

Constructor for the HedString class.

**Parameters**

- **hed_string** (*str*) – A HED string consisting of tags and tag groups.
- **hed_schema** (HedSchema) – The schema to use to identify tags.
- **def_dict** (DefinitionDict *or None*) – The def dict to use to identify def/def expand tags.

- **_contents** (*[HedGroup and/or HedTag] or None*) – Create a HedString from this exact list of children. Does not make a copy.

### Notes

- The HedString object parses its component tags and groups into a tree-like structure.

### Methods

| | |
|---|---|
| *__init__*(hed_string, hed_schema[, def_dict, ...]) | Constructor for the HedString class. |
| *append*(tag_or_group) | Add a tag or group to this group. |
| *check_if_in_original*(tag_or_group) | Check if the tag or group in original string. |
| *copy*() | Return a deep copy of this string. |
| *expand_defs*() | Replace def tags with def-expand tags |
| *find_def_tags*([recursive, include_groups]) | Find def and def-expand tags |
| *find_exact_tags*(exact_tags[, recursive, ...]) | Find the given tags. |
| *find_placeholder_tag*() | Return a placeholder tag, if present in this group. |
| *find_tags*(search_tags[, recursive, ...]) | Find the base tags and their containing groups. |
| *find_tags_with_term*(term[, recursive, ...]) | Find any tags that contain the given term. |
| *find_top_level_tags*(anchor_tags[, ...]) | Find top level groups with an anchor tag. |
| *find_wildcard_tags*(search_tags[, recursive, ...]) | Find the tags and their containing groups. |
| *from_hed_strings*(hed_strings) | Factory for creating HedStrings via combination. |
| *get_all_groups*([also_return_depth]) | Return HedGroups, including descendants and self. |
| *get_all_tags*() | Return HedTags, including descendants. |
| *get_as_form*(tag_attribute) | Get the string corresponding to the specified form. |
| *get_as_long*() | Return this HedGroup as a long tag string. |
| *get_as_original*() | Return the original form of this string. |
| *get_as_short*() | Return this HedGroup as a short tag string. |
| *get_first_group*() | Returns the first group in this hed string or group. |
| *get_original_hed_string*() | Get the original hed string. |
| *groups*() | Return the direct child groups of this group. |
| *lower*() | Convenience function, equivalent to str(self).lower() |
| *remove*(items_to_remove) | Remove any tags/groups in items_to_remove. |
| *remove_definitions*() | Remove definition tags and groups from this string. |
| *remove_refs*() | This removes any refs(tags contained entirely inside curly braces) from the string. |
| *replace*(item_to_replace, new_contents) | Replace an existing tag or group. |
| *shrink_defs*() | Replace def-expand tags with def tags |
| *sort*() | Sort the tags and groups in this HedString in a consistent order. |
| *sorted*() | Returns a sorted copy of this hed group |
| *split_hed_string*(hed_string) | Split a HED string into delimiters and tags. |
| *split_into_groups*(hed_string, hed_schema[, ...]) | Split the HED string into a parse tree. |
| *tags*() | Return the direct child tags of this group. |
| *validate*([allow_placeholders, error_handler]) | Validate the string using the schema |

### Attributes

| | |
|---|---|
| CLOSING_GROUP_CHARACTER | |
| OPENING_GROUP_CHARACTER | |
| *is_group* | Always False since the underlying string is not a group with parentheses. |
| *span* | Return the source span. |

**append**(*tag_or_group*)

Add a tag or group to this group.

**Parameters**
**tag_or_group** (HedTag *or* HedGroup) – The new object to add to this group.

**check_if_in_original**(*tag_or_group*)

Check if the tag or group in original string.

**Parameters**
**tag_or_group** (HedTag *or* HedGroup) – The HedTag or HedGroup to be looked for in this group.

**Returns**
True if in this group.

**Return type**
bool

**copy**()

Return a deep copy of this string.

**Returns**
The copied group.

**Return type**
*HedString*

**expand_defs**()

Replace def tags with def-expand tags

This does very minimal validation

**Returns**
self

**find_def_tags**(*recursive=False*, *include_groups=3*)

Find def and def-expand tags

**Parameters**

- **recursive** (*bool*) – If true, also check subgroups.

- **include_groups** (*int, 0, 1, 2, 3*) – options for return values If 0: Return only def and def expand tags/. If 1: Return only def tags and def-expand groups. If 2: Return only groups containing defs, or def-expand groups. If 3 or any other value: Return all 3 as a tuple.

**Returns**
A list of tuples. The contents depend on the values of the include_group.

**Return type**
list

**find_exact_tags**(*exact_tags*, *recursive=False*, *include_groups=1*)

Find the given tags. This will only find complete matches, any extension or value must also match.

**Parameters**

- **exact_tags** (*list of* HedTag) – A container of tags to locate.
- **recursive** (*bool*) – If true, also check subgroups.
- **include_groups** (*bool*) – 0, 1 or 2 If 0: Return only tags If 1: Return only groups If 2 or any other value: Return both

**Returns**
A list of tuples. The contents depend on the values of the include_group.

**Return type**
list

**find_placeholder_tag**()

Return a placeholder tag, if present in this group.

**Returns**
The placeholder tag if found.

**Return type**
*HedTag* or None

### Notes

- Assumes a valid HedString with no erroneous "#" characters.

**find_tags**(*search_tags*, *recursive=False*, *include_groups=2*)

Find the base tags and their containing groups. This searches by short_base_tag, ignoring any ancestors or extensions/values.

**Parameters**

- **search_tags** (*container*) – A container of short_base_tags to locate
- **recursive** (*bool*) – If true, also check subgroups.
- **include_groups** (*0, 1 or 2*) – Specify return values. If 0: return a list of the HedTags. If 1: return a list of the HedGroups containing the HedTags. If 2: return a list of tuples (HedTag, HedGroup) for the found tags.

**Returns**
The contents of the list depends on the value of include_groups.

**Return type**
list

find_tags_with_term(*term*, *recursive=False*, *include_groups=2*)

Find any tags that contain the given term.

Note: This can only find identified tags.

**Parameters**

- **term** (*str*) – A single term to search for.
- **recursive** (*bool*) – If true, recursively check subgroups.
- **include_groups** (*0, 1 or 2*) – Controls return values If 0: Return only tags If 1: Return only groups If 2 or any other value: Return both

**Return type**
list

find_top_level_tags(*anchor_tags*, *include_groups=2*)

Find top level groups with an anchor tag.

A max of 1 tag located per top level group.

**Parameters**

- **anchor_tags** (*container*) – A list/set/etc of short_base_tags to find groups by.
- **include_groups** (*0, 1 or 2*) – Parameter indicating what return values to include. If 0: return only tags. If 1: return only groups. If 2 or any other value: return both.

**Returns**
The returned result depends on include_groups:

**Return type**
list or tuple

find_wildcard_tags(*search_tags*, *recursive=False*, *include_groups=2*)

Find the tags and their containing groups.

This searches tag.short_tag, with an implicit wildcard on the end.

e.g. "Eve" will find Event, but not Sensory-event

**Parameters**

- **search_tags** (*container*) – A container of the starts of short tags to search.
- **recursive** (*bool*) – If true, also check subgroups.
- **include_groups** (*0, 1 or 2*) – Specify return values. If 0: return a list of the HedTags. If 1: return a list of the HedGroups containing the HedTags. If 2: return a list of tuples (HedTag, HedGroup) for the found tags.

**Returns**
The contents of the list depends on the value of include_groups.

**Return type**
list

**classmethod from_hed_strings**(*hed_strings*)

Factory for creating HedStrings via combination.

**Parameters**
**hed_strings** (*list or None*) – A list of HedString objects to combine. This takes ownership of their children.

**Returns**
The newly combined HedString

**Return type**
new_string(*HedString*)

**get_all_groups**(*also_return_depth=False*)

Return HedGroups, including descendants and self.

**Parameters**
**also_return_depth** (*bool*) – If True, yield tuples (group, depth) rather than just groups.

**Returns**
The list of all HedGroups in this group, including descendants and self.

**Return type**
list

**get_all_tags**()

Return HedTags, including descendants.

**Returns**
A list of all the tags in this group including descendants.

**Return type**
list

**get_as_form**(*tag_attribute*)

Get the string corresponding to the specified form.

**Parameters**
**tag_attribute** (*str*) – The hed_tag property to use to construct the string (usually short_tag or long_tag).

**Returns**
The constructed string after transformation

**Return type**
str

**get_as_long**()

Return this HedGroup as a long tag string.

**Returns**
The group as a string with all tags as long tags.

**Return type**
str

**get_as_original**()

Return the original form of this string.

**Returns**
The string with all the tags in their original form.

**Return type**
str

### Notes

Potentially with some extraneous spaces removed on returned string.

`get_as_short()`

Return this HedGroup as a short tag string.

**Returns**
The group as a string with all tags as short tags.

**Return type**
str

`get_first_group()`

Returns the first group in this hed string or group.

Useful for things like Def-expand where they only have a single group.

Raises a ValueError if there are no groups.

**Returns**
The first group

**Return type**
*HedGroup*

`get_original_hed_string()`

Get the original hed string.

**Returns**
The original string with no modification.

**Return type**
str

`groups()`

Return the direct child groups of this group.

**Returns**
All groups directly in this group, filtering out HedTag children.

**Return type**
list

`property is_group`

Always False since the underlying string is not a group with parentheses.

`lower()`

Convenience function, equivalent to str(self).lower()

`remove(`*items_to_remove: Iterable[Union[*HedTag*, *HedGroup*]]*`)`

Remove any tags/groups in items_to_remove.

**Parameters**
**items_to_remove** (*list*) – List of HedGroups and/or HedTags to remove by identity.

### Notes

- Any groups that become empty will also be pruned.
- If you pass a child and parent group, the child will also be removed from the parent.

**remove_definitions**()

Remove definition tags and groups from this string.

This does not validate definitions and will blindly removing invalid ones as well.

**remove_refs**()

This removes any refs(tags contained entirely inside curly braces) from the string.

This does NOT validate the contents of the curly braces. This is only relevant when directly editing sidecar strings. Tools will naturally ignore these.

**static replace**(*item_to_replace*, *new_contents*)

Replace an existing tag or group.

Note: This is a static method that relies on the parent attribute of item_to_replace.

**Parameters**

- **item_to_replace** (HedTag *or* HedGroup) – The item to replace must exist or this will raise an error.
- **new_contents** (HedTag *or* HedGroup) – Replacement contents.

**Raises**

- **KeyError** –
  - item_to_replace does not exist
- **AttributeError** –
  - item_to_replace has no parent set

**shrink_defs**()

Replace def-expand tags with def tags

This does not validate them and will blindly shrink invalid ones as well.

**Returns**
self

**sort**()

Sort the tags and groups in this HedString in a consistent order.

**sorted**()

Returns a sorted copy of this hed group

**Returns**
The sorted copy

**Return type**
sorted_copy (*HedGroup*)

property span

Return the source span.

**Returns**
start index of the group (including parentheses) from the source string. int: end index of the group (including parentheses) from the source string.

**Return type**
int

static split_hed_string(*hed_string*)

Split a HED string into delimiters and tags.

**Parameters**
**hed_string** (*str*) – The HED string to split.

**Returns**
A list of tuples where each tuple is (is_hed_tag, (start_pos, end_pos)).

**Return type**
list

### Notes

- **The tuple format is as follows**
  - is_hed_tag (bool): A (possible) hed tag if true, delimiter if not.
  - start_pos (int): Index of start of string in hed_string.
  - end_pos (int): Index of end of string in hed_string
- This function does not validate tags or delimiters in any form.

static split_into_groups(*hed_string*, *hed_schema*, *def_dict=None*)

Split the HED string into a parse tree.

**Parameters**
- **hed_string** (*str*) – A hed string consisting of tags and tag groups to be processed.
- **hed_schema** (HedSchema) – HED schema to use to identify tags.
- **def_dict** (DefinitionDict) – The definitions to identify

**Returns**
A list of HedTag and/or HedGroup.

**Return type**
list

**Raises**
**ValueError** –
- The string is significantly malformed, such as mismatched parentheses.

#### Notes

- The parse tree consists of tag groups, tags, and delimiters.

**tags**()

Return the direct child tags of this group.

**Returns**
All tags directly in this group, filtering out HedGroup children.

**Return type**
list

**validate**(*allow_placeholders=True*, *error_handler=None*)

Validate the string using the schema

**Parameters**

- **allow_placeholders** (*bool*) – allow placeholders in the string
- **error_handler** (ErrorHandler *or None*) – the error handler to use, creates a default one if none passed

**Returns**
A list of issues for hed string

**Return type**
issues (list of dict)

### 3.2.11 hed.models.hed_tag

#### Classes

| | |
|---|---|
| *HedTag*(hed_string, hed_schema[, span, def_dict]) | A single HED tag. |

#### 3.2.11.1 hed.models.hed_tag.HedTag

**class HedTag**(*hed_string*, *hed_schema*, *span=None*, *def_dict=None*)

Bases: object

A single HED tag.

#### Notes

- HedTag is a smart class in that it keeps track of its original value and positioning as well as pointers to the relevant HED schema information, if relevant.

**__init__**(*hed_string*, *hed_schema*, *span=None*, *def_dict=None*)

Creates a HedTag.

**Parameters**

- **hed_string** (*str*) – Source hed string for this tag.
- **hed_schema** (HedSchema) – A parameter for calculating canonical forms on creation.

- **span** (*int, int*) – The start and end indexes of the tag in the hed_string.
- **def_dict** (DefinitionDict *or None*) – The def dict to use to identify def/def expand tags.

### Methods

| | |
|---|---|
| *__init__*(hed_string, hed_schema[, span, ...]) | Creates a HedTag. |
| *base_tag_has_attribute*(tag_attribute) | Check to see if the tag has a specific attribute. |
| *copy*() | Return a deep copy of this tag. |
| *get_stripped_unit_value*() | Return the extension divided into value and units, if the units are valid. |
| *get_tag_unit_class_units*() | Get the unit class units associated with a particular tag. |
| *has_attribute*(attribute) | Return true if this is an attribute this tag has. |
| *is_basic_tag*() | Return True if a known tag with no extension or value. |
| *is_column_ref*() | Returns if this tag is a column reference from a sidecar. |
| is_placeholder() | |
| *is_takes_value_tag*() | Return true if this is a takes value tag. |
| *is_unit_class_tag*() | Return true if this is a unit class tag. |
| *is_value_class_tag*() | Return true if this is a value class tag. |
| *lower*() | Convenience function, equivalent to str(self).lower(). |
| *replace_placeholder*(placeholder_value) | If tag has a placeholder character(#), replace with value. |
| *tag_exists_in_schema*() | Get the schema entry for this tag. |
| *tag_modified*() | Return true if tag has been modified from original. |
| *value_as_default_unit*() | Returns the value converted to default units if possible. |

### Attributes

| | |
|---|---|
| *attributes* | Return a dict of all the attributes this tag has. |
| *base_tag* | Long form without value or extension. |
| *default_unit* | Get the default unit class unit for this tag. |
| *expandable* | Returns what this expands to |
| *expanded* | Returns if this is currently expanded or not. |
| *extension* | Get the extension or value of tag |
| *long_tag* | Long form including value or extension. |
| *org_base_tag* | Original form without value or extension. |
| *org_tag* | Return the original unmodified tag. |
| *schema_namespace* | Library namespace for this tag if one exists. |
| *short_base_tag* | Short form without value or extension |
| *short_tag* | Short form including value or extension. |
| *tag* | Returns the tag. |
| *unit_classes* | Return a dict of all the unit classes this tag accepts. |
| *value_classes* | Return a dict of all the value classes this tag accepts. |

**property attributes**

Return a dict of all the attributes this tag has.

Returns empty dict if this is not a value tag.

**Returns**
A dict of attributes this tag has.

**Return type**
dict

### Notes

- Returns empty dict if this is not a unit class tag.
- The dictionary has unit name as the key and HedSchemaEntry as value.

**property base_tag**

Long form without value or extension.

**Returns**
The long form of the tag, without value or extension.

**Return type**
base_tag (str)

**base_tag_has_attribute**(*tag_attribute*)

Check to see if the tag has a specific attribute.

This is primarily used to check for things like TopLevelTag on Definitions and similar.

**Parameters**
**tag_attribute** (*str*) – A tag attribute.

**Returns**
True if the tag has the specified attribute. False, if otherwise.

**Return type**
bool

**copy**()

Return a deep copy of this tag.

**Returns**
The copied group.

**Return type**
*HedTag*

**property default_unit**

Get the default unit class unit for this tag.

Only a tag with a single unit class can have default units.

**Returns**
the default unit entry for this tag, or None

**Return type**
unit(*UnitEntry* or None)

**property expandable**

Returns what this expands to

This is primarily used for Def/Def-expand tags at present.

**Returns**
Returns the expanded form of this tag

**Return type**
*HedGroup* or *HedTag* or None

**property expanded**

Returns if this is currently expanded or not.

Will always be false unless expandable is set. This is primarily used for Def/Def-expand tags at present.

**Returns**
Returns true if this is currently expanded

**Return type**
bool

**property extension**

Get the extension or value of tag

Generally this is just the portion after the last slash. Returns an empty string if no extension or value.

**Returns**
The tag name.

**Return type**
str

### Notes

- This tag must have been computed first.

**get_stripped_unit_value()**

Return the extension divided into value and units, if the units are valid.

**Returns**
The extension portion with the units removed. unit (str or None): None if no valid unit found.

**Return type**
stripped_unit_value (str)

### Examples

'Duration/3 ms' will return '3'

**get_tag_unit_class_units**()

Get the unit class units associated with a particular tag.

**Returns**
A list containing the unit class units associated with a particular tag or an empty list.

**Return type**
list

**has_attribute**(*attribute*)

Return true if this is an attribute this tag has.

**Parameters**
**attribute** (*str*) – Name of the attribute.

**Returns**
True if this tag has the attribute.

**Return type**
bool

**is_basic_tag**()

Return True if a known tag with no extension or value.

**Returns**
True if this is a known tag without extension or value.

**Return type**
bool

**is_column_ref**()

Returns if this tag is a column reference from a sidecar.

You should only see these if you are directly accessing sidecar strings, tools should remove them otherwise.

**Returns**
Returns True if this is a column ref

**Return type**
bool

**is_takes_value_tag**()

Return true if this is a takes value tag.

**Returns**
True if this is a takes value tag.

**Return type**
bool

**is_unit_class_tag**()

Return true if this is a unit class tag.

**Returns**
True if this is a unit class tag.

**Return type**
bool

`is_value_class_tag`()

Return true if this is a value class tag.

**Returns**
True if this is a tag with a value class.

**Return type**
bool

`property long_tag`

Long form including value or extension.

**Returns**
The long form of this tag.

**Return type**
str

`lower`()

Convenience function, equivalent to str(self).lower().

`property org_base_tag`

Original form without value or extension.

**Returns**
The original form of the tag, without value or extension.

**Return type**
base_tag (str)

### Notes

- Warning: This could be empty if the original tag had a name_prefix prepended. e.g. a column where "Label/" is prepended, thus the column value has zero base portion.

`property org_tag`

Return the original unmodified tag.

**Returns**
The original unmodified tag.

**Return type**
str

`replace_placeholder`(*placeholder_value*)

If tag has a placeholder character(#), replace with value.

**Parameters**
**placeholder_value** (*str*) – Value to replace placeholder with.

`property schema_namespace`

Library namespace for this tag if one exists.

**Returns**
The library namespace, including the colon.

**Return type**
namespace (str)

`property short_base_tag`

Short form without value or extension

**Returns**
The short non-extension port of a tag.

**Return type**
base_tag (str)

### Notes

- ParentNodes/Def/DefName would return just "Def".

`property short_tag`

Short form including value or extension.

**Returns**
The short form of the tag, including value or extension.

**Return type**
short_tag (str)

`property tag`

Returns the tag.

Returns the original tag if no user form set.

**Returns**
The custom set user form of the tag.

**Return type**
tag (str)

`tag_exists_in_schema()`

Get the schema entry for this tag.

**Returns**
True if this tag exists.

**Return type**
bool

### Notes

- This does NOT assure this is a valid tag.

`tag_modified()`

Return true if tag has been modified from original.

**Returns**
Return True if the tag is modified.

**Return type**
bool

### Notes

- Modifications can include adding a column name_prefix.

**property unit_classes**

Return a dict of all the unit classes this tag accepts.

**Returns**
A dict of unit classes this tag accepts.

**Return type**
unit_classes (dict)

### Notes

- Returns empty dict if this is not a unit class tag.
- The dictionary has unit name as the key and HedSchemaEntry as value.

**value_as_default_unit()**

Returns the value converted to default units if possible.

Returns None if the units are invalid.(No default unit or invalid)

**Returns**

**The extension value as default units.**
If there are not default units, returns None.

**Return type**
value (float or None)

### Examples

'Duration/300 ms' will return .3

**property value_classes**

Return a dict of all the value classes this tag accepts.

**Returns**
A dictionary of HedSchemaEntry value classes this tag accepts.

**Return type**
dict

### Notes

- Returns empty dict if this is not a value class.
- The dictionary has unit name as the key and HedSchemaEntry as value.

### 3.2.12 hed.models.indexed_df

#### Classes

| | |
|---|---|
| *IndexedDF*(tabular_input, sidecar, hed_schema) | |

#### 3.2.12.1 hed.models.indexed_df.IndexedDF

**class IndexedDF**(*tabular_input*, *sidecar*, *hed_schema*)

Bases: `object`

**__init__**(*tabular_input*, *sidecar*, *hed_schema*)

##### Methods

| | |
|---|---|
| *__init__*(tabular_input, sidecar, hed_schema) | |

### 3.2.13 hed.models.model_constants

#### Classes

| | |
|---|---|
| *DefTagNames*() | Source names for definitions, def labels, and expanded labels |

#### 3.2.13.1 hed.models.model_constants.DefTagNames

**class DefTagNames**

Bases: `object`

Source names for definitions, def labels, and expanded labels

**__init__**()

##### Methods

| | |
|---|---|
| *__init__*() | |

#### Attributes

| |
|---|
| DEFINITION_KEY |
| DEFINITION_ORG_KEY |
| DEF_EXPAND_KEY |
| DEF_EXPAND_ORG_KEY |
| DEF_KEY |
| DEF_KEYS |
| DEF_ORG_KEY |
| INSET_KEY |
| INSET_ORG_KEY |
| OFFSET_KEY |
| OFFSET_ORG_KEY |
| ONSET_KEY |
| ONSET_ORG_KEY |
| TEMPORAL_KEYS |

### 3.2.14 hed.models.sidecar

#### Classes

| | |
|---|---|
| *Sidecar*(files[, name]) | Contents of a JSON file or merged file. |

#### 3.2.14.1 hed.models.sidecar.Sidecar

**class Sidecar**(*files*, *name=None*)

Bases: `object`

Contents of a JSON file or merged file.

**__init__**(*files*, *name=None*)

Construct a Sidecar object representing a JSON file.

**Parameters**

- **files** (*str or FileLike or list*) – A string or file-like object representing a JSON file, or a list of such.

- **name** (*str or None*) – Optional name identifying this sidecar, generally a filename.

### Methods

| | |
|---|---|
| *__init__*(files[, name]) | Construct a Sidecar object representing a JSON file. |
| *extract_definitions*(hed_schema[, error_handler]) | Gather and validate definitions in metadata. |
| *get_as_json_string*() | Return this sidecar's column metadata as a string. |
| *get_column_refs*() | Returns a list of column refs found in this sidecar. |
| *get_def_dict*(hed_schema[, extra_def_dicts]) | Returns the definition dict for this sidecar. |
| *load_sidecar_file*(file) | Load column metadata from a given json file. |
| *load_sidecar_files*(files) | Load json from a given file or list |
| *save_as_json*(save_filename) | Save column metadata to a JSON file. |
| *validate*(hed_schema[, extra_def_dicts, ...]) | Create a SidecarValidator and validate this sidecar with the schema. |

### Attributes

| | |
|---|---|
| *all_hed_columns* | Returns all columns that are HED compatible |
| *column_data* | Generates the ColumnMetadata for this sidecar |
| *def_dict* | This is the definitions from this sidecar. |

**property all_hed_columns**

Returns all columns that are HED compatible

**Returns**
A list of all valid hed columns by name

**Return type**
column_refs(list)

**property column_data**

Generates the ColumnMetadata for this sidecar

**Returns**
ColumnMetadata}): the column metadata defined by this sidecar

**Return type**
dict({str

**property def_dict**

This is the definitions from this sidecar.

Generally you should instead call get_def_dict to get the relevant definitions

**Returns**
The definitions for this sidecar

**Return type**
*DefinitionDict*

**extract_definitions**(*hed_schema*, *error_handler=None*)

Gather and validate definitions in metadata.

**Parameters**

- **hed_schema** (HedSchema) – The schema to used to identify tags.
- **error_handler** (ErrorHandler *or None*) – The error handler to use for context, uses a default one if None.

**Returns**
Contains all the definitions located in the sidecar.

**Return type**
*DefinitionDict*

**get_as_json_string**()

Return this sidecar's column metadata as a string.

**Returns**
The json string representing this sidecar.

**Return type**
str

**get_column_refs**()

Returns a list of column refs found in this sidecar.

This does not validate

**Returns**
A list of unique column refs found

**Return type**
column_refs(list)

**get_def_dict**(*hed_schema*, *extra_def_dicts=None*)

Returns the definition dict for this sidecar.

**Parameters**

- **hed_schema** (HedSchema) – used to identify tags to find definitions
- **extra_def_dicts** (*list*, DefinitionDict, *or None*) – Extra dicts to add to the list.

**Returns**
A single definition dict representing all the data(and extra def dicts)

**Return type**
*DefinitionDict*

**load_sidecar_file**(*file*)

Load column metadata from a given json file.

**Parameters**
**file** (*str or FileLike*) – If a string, this is a filename. Otherwise, it will be parsed as a file-like.

**Raises**
***HedFileError*** –

- If the file was not found or could not be parsed into JSON.

**load_sidecar_files**(*files*)

Load json from a given file or list

**Parameters**

**files** (*str or FileLike or list*) – A string or file-like object representing a JSON file, or a list of such.

**Raises**

*HedFileError* –

- If the file was not found or could not be parsed into JSON.

**save_as_json**(*save_filename*)

Save column metadata to a JSON file.

**Parameters**

**save_filename** (*str*) – Path to save file

**validate**(*hed_schema*, *extra_def_dicts=None*, *name=None*, *error_handler=None*)

Create a SidecarValidator and validate this sidecar with the schema.

**Parameters**

- **hed_schema** (HedSchema) – Input data to be validated.
- **extra_def_dicts** (*list or* DefinitionDict) – Extra def dicts in addition to sidecar.
- **name** (*str*) – The name to report this sidecar as.
- **error_handler** (ErrorHandler) – Error context to use. Creates a new one if None.

**Returns**

A list of issues associated with each level in the HED string.

**Return type**

issues (list of dict)

### 3.2.15 hed.models.spreadsheet_input

#### Classes

| | |
|---|---|
| *SpreadsheetInput*([file, file_type, ...]) | A spreadsheet of HED tags. |

#### 3.2.15.1 hed.models.spreadsheet_input.SpreadsheetInput

**class SpreadsheetInput**(*file=None*, *file_type=None*, *worksheet_name=None*, *tag_columns=None*, *has_column_names=True*, *column_prefix_dictionary=None*, *name=None*)

Bases: *BaseInput*

A spreadsheet of HED tags.

**__init__**(*file=None*, *file_type=None*, *worksheet_name=None*, *tag_columns=None*, *has_column_names=True*, *column_prefix_dictionary=None*, *name=None*)

Constructor for the SpreadsheetInput class.

**Parameters**

- **file** (*str or file like*) – An xlsx/tsv file to open or a File object.

- **file_type** (*str or None*) – ".xlsx" for excel, ".tsv" or ".txt" for tsv. data. If file is a string, the
- **worksheet_name** (*str or None*) – The name of the Excel workbook worksheet that contains the HED tags. Not applicable to tsv files. If omitted for Excel, the first worksheet is assumed.
- **tag_columns** (*list*) – A list of ints containing the columns that contain the HED tags. The default value is [1] indicating only the second column has tags.
- **has_column_names** (*bool*) – True if file has column names. Validation will skip over the first line of the file if the spreadsheet as column names.
- **column_prefix_dictionary** (*dict*) – Dictionary with keys that are column numbers/names and values are HED tag prefixes to prepend to the tags in that column before processing.

### Notes

- column_prefix_dictionary may be deprecated/renamed. These are no longer prefixes, but rather converted to value columns. eg. {"key": "Description", 1: "Label/"} will turn into value columns as {"key": "Description/#", 1: "Label/#"} It will be a validation issue if column 1 is called "key" in the above example. This means it no longer accepts anything but the value portion only in the columns.

**Raises**

- *HedFileError* –
  - file is blank
  - An invalid dataframe was passed with size 0
  - An invalid extension was provided
  - A duplicate or empty column name appears
- **OSError** –
  - Cannot open the indicated file
- **KeyError** –
  - The specified worksheet name does not exist

### Methods

| | |
|---|---|
| *__init__*([file, file_type, worksheet_name, ...]) | Constructor for the SpreadsheetInput class. |
| *assemble*([mapper, skip_curly_braces]) | Assembles the hed strings |
| *column_metadata*() | Get the metadata for each column |
| *combine_dataframe*(dataframe) | Combines all columns in the given dataframe into a single HED string series, |
| *convert_to_form*(hed_schema, tag_form) | Convert all tags in underlying dataframe to the specified form. |
| *convert_to_long*(hed_schema) | Convert all tags in underlying dataframe to long form. |
| *convert_to_short*(hed_schema) | Convert all tags in underlying dataframe to short form. |
| *expand_defs*(hed_schema, def_dict) | Shrinks any def-expand found in the underlying dataframe. |
| *get_column_refs*() | Returns a list of column refs for this file. |
| *get_def_dict*(hed_schema[, extra_def_dicts]) | Returns the definition dict for this file |
| *get_worksheet*([worksheet_name]) | Get the requested worksheet. |
| *reset_mapper*(new_mapper) | Set mapper to a different view of the file. |
| *set_cell*(row_number, column_number, ...[, ...]) | Replace the specified cell with transformed text. |
| *shrink_defs*(hed_schema) | Shrinks any def-expand found in the underlying dataframe. |
| *to_csv*([file]) | Write to file or return as a string. |
| *to_excel*(file) | Output to an Excel file. |
| *validate*(hed_schema[, extra_def_dicts, ...]) | Creates a SpreadsheetValidator and returns all issues with this fil |

### Attributes

| |
|---|
| COMMA_DELIMITER |
| EXCEL_EXTENSION |
| FILE_EXTENSION |
| FILE_INPUT |
| STRING_INPUT |
| TAB_DELIMITER |
| TEXT_EXTENSION |

| | |
|---|---|
| *columns* | Returns a list of the column names. |
| *dataframe* | The underlying dataframe. |
| *dataframe_a* | Return the assembled dataframe |
| *has_column_names* | True if dataframe has column names. |
| *loaded_workbook* | The underlying loaded workbooks. |
| *name* | Name of the data. |
| *onsets* | Returns the onset column if it exists |
| *series_a* | Return the assembled dataframe as a series |
| *series_filtered* | Return the assembled dataframe as a series, with rows that have the same onset combined |
| *worksheet_name* | The worksheet name. |

**assemble**(*mapper=None*, *skip_curly_braces=False*)

Assembles the hed strings

**Parameters**

- **mapper** (*ColumnMapper or None*) – Generally pass none here unless you want special behavior.
- **skip_curly_braces** (*bool*) – If True, don't plug in curly brace values into columns.

**Returns**
the assembled dataframe

**Return type**
Dataframe

**column_metadata**()

Get the metadata for each column

**Returns**
number/ColumnMeta pairs

**Return type**
dict

**property columns**

Returns a list of the column names.

Empty if no column names.

**Returns**
the column names

**Return type**
columns(list)

`static combine_dataframe`(*dataframe*)

**Combines all columns in the given dataframe into a single HED string series,**
skipping empty columns and columns with empty strings.

**Parameters**
**dataframe** (*Dataframe*) – The dataframe to combine

**Returns**
the assembled series

**Return type**
Series

`convert_to_form`(*hed_schema*, *tag_form*)

Convert all tags in underlying dataframe to the specified form.

**Parameters**

- **hed_schema** (HedSchema) – The schema to use to convert tags.
- **tag_form** (*str*) – HedTag property to convert tags to. Most cases should use convert_to_short or convert_to_long below.

`convert_to_long`(*hed_schema*)

Convert all tags in underlying dataframe to long form.

**Parameters**
**hed_schema** (HedSchema *or None*) – The schema to use to convert tags.

`convert_to_short`(*hed_schema*)

Convert all tags in underlying dataframe to short form.

**Parameters**
**hed_schema** (HedSchema) – The schema to use to convert tags.

`property dataframe`

The underlying dataframe.

`property dataframe_a`

**Return the assembled dataframe**
Probably a placeholder name.

**Returns**
the assembled dataframe

**Return type**
Dataframe

`expand_defs`(*hed_schema*, *def_dict*)

Shrinks any def-expand found in the underlying dataframe.

**Parameters**

- **hed_schema** (HedSchema *or None*) – The schema to use to identify defs
- **def_dict** (DefinitionDict) – The definitions to expand

**get_column_refs**()

Returns a list of column refs for this file.

Default implementation returns none.

**Returns**
A list of unique column refs found

**Return type**
column_refs(list)

**get_def_dict**(*hed_schema*, *extra_def_dicts=None*)

Returns the definition dict for this file

Note: Baseclass implementation returns just extra_def_dicts.

**Parameters**

- **hed_schema** (HedSchema) – used to identify tags to find definitions(if needed)
- **extra_def_dicts** (*list, DefinitionDict, or None*) – Extra dicts to add to the list.

**Returns**
A single definition dict representing all the data(and extra def dicts)

**Return type**
*DefinitionDict*

**get_worksheet**(*worksheet_name=None*)

Get the requested worksheet.

**Parameters**
**worksheet_name** (*str or None*) – The name of the requested worksheet by name or the first one if None.

**Returns**
The workbook request.

**Return type**
openpyxl.workbook.Workbook

### Notes

If None, returns the first worksheet.

**Raises**
**KeyError** –

- The specified worksheet name does not exist

**property has_column_names**

True if dataframe has column names.

**property loaded_workbook**

The underlying loaded workbooks.

**property name**

Name of the data.

**property onsets**

Returns the onset column if it exists

**reset_mapper**(*new_mapper*)

Set mapper to a different view of the file.

**Parameters**

**new_mapper** (ColumnMapper) – A column mapper to be associated with this base input.

**property series_a**

Return the assembled dataframe as a series

**Returns**

the assembled dataframe with columns merged

**Return type**

Series

**property series_filtered**

Return the assembled dataframe as a series, with rows that have the same onset combined

**Returns**

the assembled dataframe with columns merged, and the rows filtered together

**Return type**

Series

**set_cell**(*row_number*, *column_number*, *new_string_obj*, *tag_form='short_tag'*)

Replace the specified cell with transformed text.

**Parameters**

- **row_number** (*int*) – The row number of the spreadsheet to set.
- **column_number** (*int*) – The column number of the spreadsheet to set.
- **new_string_obj** (HedString) – Object with text to put in the given cell.
- **tag_form** (*str*) – Version of the tags (short_tag, long_tag, base_tag, etc)

### Notes

Any attribute of a HedTag that returns a string is a valid value of tag_form.

**Raises**

- **ValueError** –
  - There is not a loaded dataframe
- **KeyError** –
  - the indicated row/column does not exist
- **AttributeError** –
  - The indicated tag_form is not an attribute of HedTag

**shrink_defs**(*hed_schema*)

Shrinks any def-expand found in the underlying dataframe.

**Parameters**
**hed_schema** (*HedSchema or None*) – The schema to use to identify defs

**to_csv**(*file=None*)

Write to file or return as a string.

**Parameters**
**file** (*str, file-like, or None*) – Location to save this file. If None, return as string.

**Returns**
None if file is given or the contents as a str if file is None.

**Return type**
None or str

**Raises**
**OSError** –

- Cannot open the indicated file

**to_excel**(*file*)

Output to an Excel file.

**Parameters**
**file** (*str or file-like*) – Location to save this base input.

**Raises**

- **ValueError** –
  - if empty file object was passed
- **OSError** –
  - Cannot open the indicated file

**validate**(*hed_schema*, *extra_def_dicts=None*, *name=None*, *error_handler=None*)

Creates a SpreadsheetValidator and returns all issues with this fil

**Parameters**

- **hed_schema** (*HedSchema*) – The schema to use for validation
- **extra_def_dicts** (*list of DefDict or DefDict*) – all definitions to use for validation
- **name** (*str*) – The name to report errors from this file as
- **error_handler** (*ErrorHandler*) – Error context to use. Creates a new one if None

**Returns**
A list of issues for hed string

**Return type**
issues (list of dict)

**property worksheet_name**

The worksheet name.

### 3.2.16 hed.models.string_util

#### Functions

| | |
|---|---|
| *gather_descriptions*(hed_string) | Removes any description tags from the string and concatenates them |
| *split_base_tags*(hed_string, base_tags[, ...]) | Splits a HedString object into two separate HedString objects based on the presence of base tags. |
| *split_def_tags*(hed_string, def_names[, ...]) | Splits a HedString object into two separate HedString objects based on the presence of wildcard tags. |

#### 3.2.16.1 hed.models.string_util.gather_descriptions

**gather_descriptions**(*hed_string*)

Removes any description tags from the string and concatenates them

**Parameters**
**hed_string** (HedString) – To be modified

**Returns: tuple**
description(str): The concatenated values of all description tags.

**Side-effect:**
The input HedString has its Definition tags removed.

#### 3.2.16.2 hed.models.string_util.split_base_tags

**split_base_tags**(*hed_string*, *base_tags*, *remove_group=False*)

Splits a HedString object into two separate HedString objects based on the presence of base tags.

**Parameters**

- **hed_string** (HedString) – The input HedString object to be split.
- **base_tags** (*list of str*) – A list of strings representing the base tags. This is matching the base tag NOT all the terms above it.
- **remove_group** (*bool, optional*) – Flag indicating whether to remove the parent group. Defaults to False.

**Returns**

**A tuple containing two HedString objects:**

- The first HedString object contains the remaining tags from hed_string.
- The second HedString object contains the tags from hed_string that match the base_tags.

**Return type**
tuple

#### 3.2.16.3 hed.models.string_util.split_def_tags

**split_def_tags**(*hed_string*, *def_names*, *remove_group=False*)

Splits a HedString object into two separate HedString objects based on the presence of wildcard tags.

This does NOT handle def-expand tags currently.

**Parameters**

- **hed_string** (HedString) – The input HedString object to be split.
- **def_names** (*list of str*) – A list of def names to search for. Can optionally include a value.
- **remove_group** (*bool, optional*) – Flag indicating whether to remove the parent group. Defaults to False.

**Returns**

**A tuple containing two HedString objects:**

- The first HedString object contains the remaining tags from hed_string.
- The second HedString object contains the tags from hed_string that match the def_names.

**Return type**
tuple

### 3.2.17 hed.models.tabular_input

#### Classes

| | |
|---|---|
| *TabularInput*([file, sidecar, name]) | A BIDS tabular tsv file with sidecar. |

#### 3.2.17.1 hed.models.tabular_input.TabularInput

**class TabularInput**(*file=None*, *sidecar=None*, *name=None*)

Bases: *BaseInput*

A BIDS tabular tsv file with sidecar.

**__init__**(*file=None*, *sidecar=None*, *name=None*)

Constructor for the TabularInput class.

**Parameters**

- **file** (*str or FileLike*) – A tsv file to open.
- **sidecar** (*str or* Sidecar *or FileLike*) – A Sidecar or source file/filename.
- **name** (*str*) – The name to display for this file for error purposes.

**Raises**

- ***HedFileError*** –
  - file is blank
  - An invalid dataframe was passed with size 0

  - An invalid extension was provided
  - A duplicate or empty column name appears
- **OSError** –
  - Cannot open the indicated file
- **ValueError** –
  - This file has no column names

### Methods

| | |
|---|---|
| *__init__*([file, sidecar, name]) | Constructor for the TabularInput class. |
| *assemble*([mapper, skip_curly_braces]) | Assembles the hed strings |
| *column_metadata*() | Get the metadata for each column |
| *combine_dataframe*(dataframe) | Combines all columns in the given dataframe into a single HED string series, |
| *convert_to_form*(hed_schema, tag_form) | Convert all tags in underlying dataframe to the specified form. |
| *convert_to_long*(hed_schema) | Convert all tags in underlying dataframe to long form. |
| *convert_to_short*(hed_schema) | Convert all tags in underlying dataframe to short form. |
| *expand_defs*(hed_schema, def_dict) | Shrinks any def-expand found in the underlying dataframe. |
| *get_column_refs*() | Returns a list of column refs for this file. |
| *get_def_dict*(hed_schema[, extra_def_dicts]) | Returns the definition dict for this sidecar. |
| *get_worksheet*([worksheet_name]) | Get the requested worksheet. |
| *reset_column_mapper*([sidecar]) | Change the sidecars and settings. |
| *reset_mapper*(new_mapper) | Set mapper to a different view of the file. |
| *set_cell*(row_number, column_number, ...[, ...]) | Replace the specified cell with transformed text. |
| *shrink_defs*(hed_schema) | Shrinks any def-expand found in the underlying dataframe. |
| *to_csv*([file]) | Write to file or return as a string. |
| *to_excel*(file) | Output to an Excel file. |
| *validate*(hed_schema[, extra_def_dicts, ...]) | Creates a SpreadsheetValidator and returns all issues with this fil |

### Attributes

| |
|---|
| COMMA_DELIMITER |
| EXCEL_EXTENSION |
| FILE_EXTENSION |
| FILE_INPUT |
| HED_COLUMN_NAME |
| STRING_INPUT |
| TAB_DELIMITER |
| TEXT_EXTENSION |

| | |
|---|---|
| *columns* | Returns a list of the column names. |
| *dataframe* | The underlying dataframe. |
| *dataframe_a* | Return the assembled dataframe |
| *has_column_names* | True if dataframe has column names. |
| *loaded_workbook* | The underlying loaded workbooks. |
| *name* | Name of the data. |
| *onsets* | Returns the onset column if it exists |
| *series_a* | Return the assembled dataframe as a series |
| *series_filtered* | Return the assembled dataframe as a series, with rows that have the same onset combined |
| *worksheet_name* | The worksheet name. |

**assemble**(*mapper=None*, *skip_curly_braces=False*)

Assembles the hed strings

**Parameters**

- **mapper** (*ColumnMapper or None*) – Generally pass none here unless you want special behavior.
- **skip_curly_braces** (*bool*) – If True, don't plug in curly brace values into columns.

**Returns**
the assembled dataframe

**Return type**
Dataframe

**column_metadata**()

Get the metadata for each column

**Returns**
number/ColumnMeta pairs

**Return type**
dict

**property columns**

Returns a list of the column names.

Empty if no column names.

**Returns**
the column names

**Return type**
columns(list)

**static combine_dataframe**(*dataframe*)

**Combines all columns in the given dataframe into a single HED string series,**
skipping empty columns and columns with empty strings.

**Parameters**
**dataframe** (*Dataframe*) – The dataframe to combine

**Returns**
the assembled series

**Return type**
Series

**convert_to_form**(*hed_schema*, *tag_form*)

Convert all tags in underlying dataframe to the specified form.

**Parameters**

- **hed_schema** (HedSchema) – The schema to use to convert tags.
- **tag_form** (*str*) – HedTag property to convert tags to. Most cases should use convert_to_short or convert_to_long below.

**convert_to_long**(*hed_schema*)

Convert all tags in underlying dataframe to long form.

**Parameters**
**hed_schema** (HedSchema *or None*) – The schema to use to convert tags.

**convert_to_short**(*hed_schema*)

Convert all tags in underlying dataframe to short form.

**Parameters**
**hed_schema** (HedSchema) – The schema to use to convert tags.

**property dataframe**

The underlying dataframe.

**property dataframe_a**

**Return the assembled dataframe**
Probably a placeholder name.

**Returns**
the assembled dataframe

**Return type**
Dataframe

**expand_defs**(*hed_schema*, *def_dict*)

Shrinks any def-expand found in the underlying dataframe.

**Parameters**

- **hed_schema** (HedSchema *or None*) – The schema to use to identify defs
- **def_dict** (DefinitionDict) – The definitions to expand

**get_column_refs**()

Returns a list of column refs for this file.

Default implementation returns none.

**Returns**
A list of unique column refs found

**Return type**
column_refs(list)

**get_def_dict**(*hed_schema*, *extra_def_dicts=None*)

Returns the definition dict for this sidecar.

**Parameters**

- **hed_schema** (HedSchema) – used to identify tags to find definitions
- **extra_def_dicts** (*list, DefinitionDict, or None*) – Extra dicts to add to the list.

**Returns**
A single definition dict representing all the data(and extra def dicts)

**Return type**
*DefinitionDict*

**get_worksheet**(*worksheet_name=None*)

Get the requested worksheet.

**Parameters**
**worksheet_name** (*str or None*) – The name of the requested worksheet by name or the first one if None.

**Returns**
The workbook request.

**Return type**
openpyxl.workbook.Workbook

### Notes

If None, returns the first worksheet.

**Raises**
**KeyError** –

- The specified worksheet name does not exist

**property has_column_names**

True if dataframe has column names.

**property loaded_workbook**

The underlying loaded workbooks.

**property name**

Name of the data.

**property onsets**

Returns the onset column if it exists

**reset_column_mapper**(*sidecar=None*)

Change the sidecars and settings.

**Parameters**

**sidecar** (*str or [str] or Sidecar or [Sidecar]*) – A list of json filenames to pull sidecar info from.

**reset_mapper**(*new_mapper*)

Set mapper to a different view of the file.

**Parameters**

**new_mapper** (ColumnMapper) – A column mapper to be associated with this base input.

**property series_a**

Return the assembled dataframe as a series

**Returns**

the assembled dataframe with columns merged

**Return type**

Series

**property series_filtered**

Return the assembled dataframe as a series, with rows that have the same onset combined

**Returns**

the assembled dataframe with columns merged, and the rows filtered together

**Return type**

Series

**set_cell**(*row_number*, *column_number*, *new_string_obj*, *tag_form='short_tag'*)

Replace the specified cell with transformed text.

**Parameters**

- **row_number** (*int*) – The row number of the spreadsheet to set.
- **column_number** (*int*) – The column number of the spreadsheet to set.
- **new_string_obj** (HedString) – Object with text to put in the given cell.
- **tag_form** (*str*) – Version of the tags (short_tag, long_tag, base_tag, etc)

### Notes

Any attribute of a HedTag that returns a string is a valid value of tag_form.

**Raises**

- **ValueError** –
  - There is not a loaded dataframe
- **KeyError** –
  - the indicated row/column does not exist
- **AttributeError** –
  - The indicated tag_form is not an attribute of HedTag

**shrink_defs**(*hed_schema*)

Shrinks any def-expand found in the underlying dataframe.

**Parameters**
**hed_schema** (*HedSchema or None*) – The schema to use to identify defs

**to_csv**(*file=None*)

Write to file or return as a string.

**Parameters**
**file** (*str, file-like, or None*) – Location to save this file. If None, return as string.

**Returns**
None if file is given or the contents as a str if file is None.

**Return type**
None or str

**Raises**
**OSError** –

- Cannot open the indicated file

**to_excel**(*file*)

Output to an Excel file.

**Parameters**
**file** (*str or file-like*) – Location to save this base input.

**Raises**

- **ValueError** –
  - if empty file object was passed
- **OSError** –
  - Cannot open the indicated file

**validate**(*hed_schema*, *extra_def_dicts=None*, *name=None*, *error_handler=None*)

Creates a SpreadsheetValidator and returns all issues with this fil

**Parameters**

- **hed_schema** (*HedSchema*) – The schema to use for validation
- **extra_def_dicts** (*list of DefDict or DefDict*) – all definitions to use for validation

- **name** (*str*) – The name to report errors from this file as
- **error_handler** (ErrorHandler) – Error context to use. Creates a new one if None

**Returns**
A list of issues for hed string

**Return type**
issues (list of dict)

**property worksheet_name**
The worksheet name.

### 3.2.18 hed.models.timeseries_input

#### Classes

| | |
|---|---|
| *TimeseriesInput*([file, sidecar, ...]) | A BIDS time series tsv file. |

#### 3.2.18.1 hed.models.timeseries_input.TimeseriesInput

**class TimeseriesInput**(*file=None*, *sidecar=None*, *extra_def_dicts=None*, *name=None*)

Bases: *BaseInput*

A BIDS time series tsv file.

**__init__**(*file=None*, *sidecar=None*, *extra_def_dicts=None*, *name=None*)
Constructor for the TimeseriesInput class.

**Parameters**

- **file** (*str or file like*) – A tsv file to open.
- **sidecar** (*str or* Sidecar) – A json sidecar to pull metadata from.
- **extra_def_dicts** (DefinitionDict, *list, or None*) – Additional definition dictionaries.
- **name** (*str*) – The name to display for this file for error purposes.

##### Notes

- The extra_def_dicts are external definitions that override the ones in the object.

### Methods

| | |
|---|---|
| *__init__*([file, sidecar, extra_def_dicts, name]) | Constructor for the TimeseriesInput class. |
| *assemble*([mapper, skip_curly_braces]) | Assembles the hed strings |
| *column_metadata*() | Get the metadata for each column |
| *combine_dataframe*(dataframe) | Combines all columns in the given dataframe into a single HED string series, |
| *convert_to_form*(hed_schema, tag_form) | Convert all tags in underlying dataframe to the specified form. |
| *convert_to_long*(hed_schema) | Convert all tags in underlying dataframe to long form. |
| *convert_to_short*(hed_schema) | Convert all tags in underlying dataframe to short form. |
| *expand_defs*(hed_schema, def_dict) | Shrinks any def-expand found in the underlying dataframe. |
| *get_column_refs*() | Returns a list of column refs for this file. |
| *get_def_dict*(hed_schema[, extra_def_dicts]) | Returns the definition dict for this file |
| *get_worksheet*([worksheet_name]) | Get the requested worksheet. |
| *reset_mapper*(new_mapper) | Set mapper to a different view of the file. |
| *set_cell*(row_number, column_number, ...[, ...]) | Replace the specified cell with transformed text. |
| *shrink_defs*(hed_schema) | Shrinks any def-expand found in the underlying dataframe. |
| *to_csv*([file]) | Write to file or return as a string. |
| *to_excel*(file) | Output to an Excel file. |
| *validate*(hed_schema[, extra_def_dicts, ...]) | Creates a SpreadsheetValidator and returns all issues with this fil |

### Attributes

| |
|---|
| COMMA_DELIMITER |
| EXCEL_EXTENSION |
| FILE_EXTENSION |
| FILE_INPUT |
| HED_COLUMN_NAME |
| STRING_INPUT |
| TAB_DELIMITER |
| TEXT_EXTENSION |

| | |
|---|---|
| *columns* | Returns a list of the column names. |
| *dataframe* | The underlying dataframe. |
| *dataframe_a* | Return the assembled dataframe |
| *has_column_names* | True if dataframe has column names. |
| *loaded_workbook* | The underlying loaded workbooks. |
| *name* | Name of the data. |
| *onsets* | Returns the onset column if it exists |
| *series_a* | Return the assembled dataframe as a series |
| *series_filtered* | Return the assembled dataframe as a series, with rows that have the same onset combined |
| *worksheet_name* | The worksheet name. |

**assemble**(*mapper=None*, *skip_curly_braces=False*)

Assembles the hed strings

**Parameters**

- **mapper** (*ColumnMapper or None*) – Generally pass none here unless you want special behavior.
- **skip_curly_braces** (*bool*) – If True, don't plug in curly brace values into columns.

**Returns**
the assembled dataframe

**Return type**
Dataframe

**column_metadata**()

Get the metadata for each column

**Returns**
number/ColumnMeta pairs

**Return type**
dict

property columns

Returns a list of the column names.

Empty if no column names.

**Returns**
the column names

**Return type**
columns(list)

static combine_dataframe(*dataframe*)

**Combines all columns in the given dataframe into a single HED string series,**
skipping empty columns and columns with empty strings.

**Parameters**
**dataframe** (*Dataframe*) – The dataframe to combine

**Returns**
the assembled series

**Return type**
Series

convert_to_form(*hed_schema*, *tag_form*)

Convert all tags in underlying dataframe to the specified form.

**Parameters**

- **hed_schema** (HedSchema) – The schema to use to convert tags.
- **tag_form** (*str*) – HedTag property to convert tags to. Most cases should use convert_to_short or convert_to_long below.

convert_to_long(*hed_schema*)

Convert all tags in underlying dataframe to long form.

**Parameters**
**hed_schema** (HedSchema *or None*) – The schema to use to convert tags.

convert_to_short(*hed_schema*)

Convert all tags in underlying dataframe to short form.

**Parameters**
**hed_schema** (HedSchema) – The schema to use to convert tags.

property dataframe

The underlying dataframe.

property dataframe_a

**Return the assembled dataframe**
Probably a placeholder name.

**Returns**
the assembled dataframe

**Return type**
Dataframe

**expand_defs**(*hed_schema*, *def_dict*)

Shrinks any def-expand found in the underlying dataframe.

**Parameters**

- **hed_schema** (*HedSchema or None*) – The schema to use to identify defs
- **def_dict** (*DefinitionDict*) – The definitions to expand

**get_column_refs**()

Returns a list of column refs for this file.

Default implementation returns none.

**Returns**
A list of unique column refs found

**Return type**
column_refs(list)

**get_def_dict**(*hed_schema*, *extra_def_dicts=None*)

Returns the definition dict for this file

Note: Baseclass implementation returns just extra_def_dicts.

**Parameters**

- **hed_schema** (*HedSchema*) – used to identify tags to find definitions(if needed)
- **extra_def_dicts** (*list, DefinitionDict, or None*) – Extra dicts to add to the list.

**Returns**
A single definition dict representing all the data(and extra def dicts)

**Return type**
*DefinitionDict*

**get_worksheet**(*worksheet_name=None*)

Get the requested worksheet.

**Parameters**
**worksheet_name** (*str or None*) – The name of the requested worksheet by name or the first one if None.

**Returns**
The workbook request.

**Return type**
openpyxl.workbook.Workbook

### Notes

If None, returns the first worksheet.

**Raises**
**KeyError** –

- The specified worksheet name does not exist

**property has_column_names**

True if dataframe has column names.

**property loaded_workbook**

The underlying loaded workbooks.

**property name**

Name of the data.

**property onsets**

Returns the onset column if it exists

**reset_mapper**(*new_mapper*)

Set mapper to a different view of the file.

**Parameters**

**new_mapper** (ColumnMapper) – A column mapper to be associated with this base input.

**property series_a**

Return the assembled dataframe as a series

**Returns**

the assembled dataframe with columns merged

**Return type**

Series

**property series_filtered**

Return the assembled dataframe as a series, with rows that have the same onset combined

**Returns**

the assembled dataframe with columns merged, and the rows filtered together

**Return type**

Series

**set_cell**(*row_number*, *column_number*, *new_string_obj*, *tag_form='short_tag'*)

Replace the specified cell with transformed text.

**Parameters**

- **row_number** (*int*) – The row number of the spreadsheet to set.
- **column_number** (*int*) – The column number of the spreadsheet to set.
- **new_string_obj** (HedString) – Object with text to put in the given cell.
- **tag_form** (*str*) – Version of the tags (short_tag, long_tag, base_tag, etc)

### Notes

Any attribute of a HedTag that returns a string is a valid value of tag_form.

**Raises**

- **ValueError** –
  - There is not a loaded dataframe
- **KeyError** –
  - the indicated row/column does not exist
- **AttributeError** –
  - The indicated tag_form is not an attribute of HedTag

**shrink_defs**(*hed_schema*)

Shrinks any def-expand found in the underlying dataframe.

**Parameters**
**hed_schema** (HedSchema *or None*) – The schema to use to identify defs

**to_csv**(*file=None*)

Write to file or return as a string.

**Parameters**
**file** (*str, file-like, or None*) – Location to save this file. If None, return as string.

**Returns**
None if file is given or the contents as a str if file is None.

**Return type**
None or str

**Raises**
**OSError** –

- Cannot open the indicated file

**to_excel**(*file*)

Output to an Excel file.

**Parameters**
**file** (*str or file-like*) – Location to save this base input.

**Raises**

- **ValueError** –
  - if empty file object was passed
- **OSError** –
  - Cannot open the indicated file

**validate**(*hed_schema*, *extra_def_dicts=None*, *name=None*, *error_handler=None*)

Creates a SpreadsheetValidator and returns all issues with this fil

**Parameters**

- **hed_schema** (HedSchema) – The schema to use for validation
- **extra_def_dicts** (*list of DefDict or DefDict*) – all definitions to use for validation
- **name** (*str*) – The name to report errors from this file as
- **error_handler** (ErrorHandler) – Error context to use. Creates a new one if None

**Returns**
A list of issues for hed string

**Return type**
issues (list of dict)

**property worksheet_name**

The worksheet name.

## 3.3 hed.schema

Data structures for handling the HED schema.

### Modules

| | |
|---|---|
| hed.schema.hed_cache | Infrastructure for caching HED schema from remote repositories. |
| hed.schema.hed_schema | |
| hed.schema.hed_schema_base | Abstract base class for HedSchema and HedSchemaGroup, showing the common functionality |
| hed.schema.hed_schema_constants | |
| hed.schema.hed_schema_entry | |
| hed.schema.hed_schema_group | |
| hed.schema.hed_schema_io | Utilities for loading and outputting HED schema. |
| hed.schema.hed_schema_section | |
| hed.schema.schema_attribute_validators | The built-in functions to validate known attributes. |
| hed.schema.schema_compare | |
| hed.schema.schema_compliance | Utilities for HED schema checking. |
| hed.schema.schema_io | |
| hed.schema.schema_validation_util | Utilities used in HED validation/loading using a HED schema. |

### 3.3.1 hed.schema.hed_cache

Infrastructure for caching HED schema from remote repositories.

#### Functions

| | |
|---|---|
| cache_local_versions(cache_folder) | Cache all schemas included with the hed installation. |
| cache_specific_url(hed_xml_url[, ...]) | Cache a file from a URL. |
| cache_xml_versions([hed_base_urls, ...]) | Cache all schemas at the given URLs. |
| get_cache_directory() | Return the current value of HED_CACHE_DIRECTORY. |
| get_hed_version_path([xml_version, ...]) | Get latest HED XML file path in a directory. |
| get_hed_versions([local_hed_directory, ...]) | Get the HED versions in the hed directory. |
| get_path_from_hed_version(hed_version[, ...]) | Return the HED XML file path for a version. |
| set_cache_directory(new_cache_dir) | Set default global hed cache directory. |

#### 3.3.1.1 hed.schema.hed_cache.cache_local_versions

**cache_local_versions**(*cache_folder*)

Cache all schemas included with the hed installation.

**Parameters**
**cache_folder** (*str*) – The folder holding the cache.

**Returns**
Returns -1 on cache access failure. None otherwise

**Return type**
int or None

#### 3.3.1.2 hed.schema.hed_cache.cache_specific_url

**cache_specific_url**(*hed_xml_url*, *xml_version=None*, *library_name=None*, *cache_folder=None*)

Cache a file from a URL.

**Parameters**

- **hed_xml_url** (*str*) – Path to an exact file at a URL, or a GitHub API url to a directory.
- **xml_version** (*str*) – If not None and hed_xml_url is a directory, return this version or None.
- **library_name** (*str or None*) – Optional schema library name.
- **cache_folder** (*str*) – The path of the hed cache. Defaults to HED_CACHE_DIRECTORY.

**Returns**
Path to local hed XML file to use.

**Return type**
str

#### 3.3.1.3 hed.schema.hed_cache.cache_xml_versions

**cache_xml_versions**(*hed_base_urls=('https://api.github.com/repos/hed-standard/hed-schemas/contents/standard_schema/hedxml', 'https://api.github.com/repos/hed-standard/hed-schemas/contents/library_schemas')*, *skip_folders=('deprecated',)*, *cache_folder=None*)

Cache all schemas at the given URLs.

**Parameters**

- **hed_base_urls** (*str or list*) – Path or list of paths.
- **skip_folders** (*list*) – A list of subfolders to skip over when downloading.
- **cache_folder** (*str*) – The folder holding the cache.

**Returns**

**Returns -1 if cache failed, a positive number meaning time in seconds since last update**
if it didn't cache, 0 if it cached successfully this time.

**Return type**
float

#### Notes

- The Default skip_folders is 'deprecated'.
- The HED cache folder defaults to HED_CACHE_DIRECTORY.
- **The directories on Github are of the form:**
  https://api.github.com/repos/hed-standard/hed-schemas/contents/standard_schema/hedxml

#### 3.3.1.4 hed.schema.hed_cache.get_cache_directory

`get_cache_directory`()

Return the current value of HED_CACHE_DIRECTORY.

#### 3.3.1.5 hed.schema.hed_cache.get_hed_version_path

`get_hed_version_path`(*xml_version=None*, *library_name=None*, *local_hed_directory=None*)

Get latest HED XML file path in a directory. Only returns filenames that exist.

**Parameters**

- **library_name** (*str or None*) – Optional the schema library name.
- **xml_version** (*str or None*) – If not None, return this version or None.
- **local_hed_directory** (*str*) – Path to local hed directory. Defaults to HED_CACHE_DIRECTORY

**Returns**

The path to the latest HED version the hed directory.

**Return type**

str

#### 3.3.1.6 hed.schema.hed_cache.get_hed_versions

`get_hed_versions`(*local_hed_directory=None*, *library_name=None*)

Get the HED versions in the hed directory.

**Parameters**

- **local_hed_directory** (*str*) – Directory to check for versions which defaults to hed_cache.
- **library_name** (*str or None*) – An optional schema library name. None retrieves the standard schema only. Pass "all" to retrieve all standard and library schemas as a dict.

**Returns**

List of version numbers or dictionary {library_name: [versions]}.

**Return type**

list or dict

#### 3.3.1.7 hed.schema.hed_cache.get_path_from_hed_version

**get_path_from_hed_version**(*hed_version*, *library_name=None*, *local_hed_directory=None*)

Return the HED XML file path for a version.

**Parameters**

- **hed_version** (*str*) – The HED version that is in the hed directory.
- **library_name** (*str or None*) – An optional schema library name.
- **local_hed_directory** (*str*) – The local hed path to use.

**Returns**

The HED XML file path in the hed directory that corresponds to the hed version specified.

**Return type**

str

##### Notes

- Note if no local directory is given, it defaults to HED_CACHE_DIRECTORY.

#### 3.3.1.8 hed.schema.hed_cache.set_cache_directory

**set_cache_directory**(*new_cache_dir*)

Set default global hed cache directory.

**Parameters**

**new_cache_dir** (*str*) – Directory to check for versions.

### 3.3.2 hed.schema.hed_schema

#### Classes

| | |
|---|---|
| *HedSchema*() | A HED schema suitable for processing. |

#### 3.3.2.1 hed.schema.hed_schema.HedSchema

**class HedSchema**

Bases: *HedSchemaBase*

A HED schema suitable for processing.

**__init__**()

Constructor for the HedSchema class.

A HedSchema can be used for validation, checking tag attributes, parsing tags, etc.

### Methods

| | |
|---|---|
| *__init__*() | Constructor for the HedSchema class. |
| *check_compliance*([check_for_warnings, name, ...]) | Check for HED3 compliance of this schema. |
| *finalize_dictionaries*() | Call to finish loading. |
| *find_tag_entry*(tag[, schema_namespace]) | Find the schema entry for a given source tag. |
| *get_all_schema_tags*([return_last_term]) | Get a list of all hed terms from the schema. |
| *get_all_tag_attributes*(tag_name[, key_class]) | Gather all attributes for a given tag name. |
| *get_as_mediawiki_string*([save_merged]) | Return the schema to a mediawiki string. |
| *get_as_xml_string*([save_merged]) | Return the schema to an XML string. |
| *get_desc_iter*() | Return an iterator over all the descriptions. |
| *get_formatted_version*() | The HED version string including namespace and library name if any of this schema. |
| *get_save_header_attributes*([save_merged]) | returns the attributes that should be saved. |
| *get_schema_versions*() | A list of HED version strings including namespace and library name if any of this schema. |
| *get_tag_attribute_names*() | Return a dict of all allowed tag attributes. |
| *get_tag_description*(tag_name[, key_class]) | Return the description associated with the tag. |
| *get_tag_entry*(name[, key_class, ...]) | Return the schema entry for this tag, if one exists. |
| *get_tags_with_attribute*(attribute[, key_class]) | Return tag entries with the given attribute. |
| *get_unknown_attributes*() | Retrieve the current list of unknown attributes. |
| *save_as_mediawiki*([filename, save_merged]) | Save as mediawiki to a file. |
| *save_as_xml*([filename, save_merged]) | Save as XML to a file. |
| *schema_for_namespace*(namespace) | Return HedSchema object for this namespace. |
| *set_schema_prefix*(schema_namespace) | Set library namespace associated for this schema. |

### Attributes

| | |
|---|---|
| *attributes* | Return the attributes schema section. |
| *library* | The name of this library schema if one exists. |
| *merged* | Returns if this schema was loaded from a merged file |
| *properties* | Return the properties schema section. |
| *tags* | Return the tag schema section. |
| *unit_classes* | Return the unit classes schema section. |
| *unit_modifiers* | Return the modifiers classes schema section |
| *valid_prefixes* | Return a list of all prefixes this schema will accept |
| *value_classes* | Return the value classes schema section. |
| *version* | The complete schema version, including prefix and library name(if applicable) |
| *version_number* | The HED version of this schema. |
| *with_standard* | The version of the base schema this is extended from, if it exists.. |

**property attributes**

Return the attributes schema section.

**Returns**

The attributes section.

**Return type**
*HedSchemaSection*

**check_compliance**(*check_for_warnings=True*, *name=None*, *error_handler=None*)

Check for HED3 compliance of this schema.

**Parameters**

- **check_for_warnings** (*bool*) – If True, checks for formatting issues like invalid characters, capitalization.
- **name** (*str*) – If present, use as the filename for context, rather than using the actual filename. Useful for temp filenames when supporting web services.
- **error_handler** (*ErrorHandler or None*) – Used to report errors. Uses a default one if none passed in.

**Returns**
A list of all warnings and errors found in the file. Each issue is a dictionary.

**Return type**
list

**finalize_dictionaries**()

Call to finish loading.

**find_tag_entry**(*tag*, *schema_namespace=''*)

Find the schema entry for a given source tag.

**Parameters**

- **tag** (*str, HedTag*) – Any form of tag to look up. Can have an extension, value, etc.
- **schema_namespace** (*str*) – The schema namespace of the tag, if any.

**Returns**
The located tag entry for this tag. str: The remainder of the tag that isn't part of the base tag. list: A list of errors while converting.

**Return type**
*HedTagEntry*

### Notes

Works left to right (which is mostly relevant for errors).

**get_all_schema_tags**(*return_last_term=False*)

Get a list of all hed terms from the schema.

**Returns**
A list of all terms(short tags) from the schema.

**Return type**
list

### Notes

Compatible with Hed2 or Hed3.

**get_all_tag_attributes**(*tag_name*, *key_class=HedSectionKey.Tags*)

Gather all attributes for a given tag name.

**Parameters**

- **tag_name** (*str*) – The name of the tag to check.
- **key_class** (*str*) – The type of attributes requested. e.g. Tag, Units, Unit modifiers, or attributes.

**Returns**
A dictionary of attribute name and attribute value.

**Return type**
dict

### Notes

If keys is None, gets all normal hed tag attributes.

**get_as_mediawiki_string**(*save_merged=False*)

Return the schema to a mediawiki string.

**Parameters**
**save_merged** (*bool*) – If true, this will save the schema as a merged schema if it is a "with-Standard" schema. If it is not a "withStandard" schema, this setting has no effect.

**Returns**
The schema as a string in mediawiki format.

**Return type**
str

**get_as_xml_string**(*save_merged=True*)

Return the schema to an XML string.

**Parameters**

- **save_merged** (*bool*) –
- **true** (*If*) –
- **schema.** (*this will save the schema as a merged schema if it is a "withStandard"*) –
- **schema** (*If it is not a "withStandard"*) –
- **effect.** (*this setting has no*) –

**Returns**
Return the schema as an XML string.

**Return type**
str

**get_desc_iter**()

Return an iterator over all the descriptions.

**Yields**
*tuple* – - str: The tag node name. - str: The description associated with the node.

`get_formatted_version()`
The HED version string including namespace and library name if any of this schema.

**Returns**
A json formatted string of the complete version of this schema including library name and namespace.

**Return type**
str

`get_save_header_attributes(save_merged=False)`
returns the attributes that should be saved.

`get_schema_versions()`
A list of HED version strings including namespace and library name if any of this schema.

**Returns**
The complete version of this schema including library name and namespace.

**Return type**
list

`get_tag_attribute_names()`
Return a dict of all allowed tag attributes.

**Returns**
A dictionary whose keys are attribute names and values are HedSchemaEntry object.

**Return type**
dict

`get_tag_description(tag_name, key_class=HedSectionKey.Tags)`
Return the description associated with the tag.

**Parameters**

- **tag_name** (*str*) – A hed tag name(or unit/unit modifier etc) with proper capitalization.
- **key_class** (*str*) – A string indicating type of description (e.g. All tags, Units, Unit modifier). The default is HedSectionKey.Tags.

**Returns**
A description of the specified tag.

**Return type**
str

`get_tag_entry(name, key_class=HedSectionKey.Tags, schema_namespace='')`
Return the schema entry for this tag, if one exists.

**Parameters**

- **name** (*str*) – Any form of basic tag(or other section entry) to look up. This will not handle extensions or similar. If this is a tag, it can have a schema namespace, but it's not required
- **key_class** (HedSectionKey *or* *str*) – The type of entry to return.
- **schema_namespace** (*str*) – Only used on Tags. If incorrect, will return None.

**Returns**
The schema entry for the given tag.

**Return type**
*HedSchemaEntry*

`get_tags_with_attribute`(*attribute*, *key_class=HedSectionKey.Tags*)

Return tag entries with the given attribute.

**Parameters**

- **attribute** (*str*) – A tag attribute. Eg HedKey.ExtensionAllowed
- **key_class** (HedSectionKey) – The HedSectionKey for the section to retrieve from.

**Returns**
A list of all tags with this attribute.

**Return type**
list

### Notes

- The result is cached so will be fast after first call.

`get_unknown_attributes`()

Retrieve the current list of unknown attributes.

**Returns**
The keys are attribute names and the values are lists of tags with this attribute.

**Return type**
dict

### Notes

- This includes attributes found in the wrong section for example unitClass attribute found on a Tag.
- The return tag list is in long form.

`property library`

The name of this library schema if one exists.

**Returns**
Library name if any.

**Return type**
str

`property merged`

Returns if this schema was loaded from a merged file

**Returns**
True if file was loaded from a merged file

**Return type**
bool

`property properties`

Return the properties schema section.

**Returns**
The properties section.

**Return type**
*HedSchemaSection*

`save_as_mediawiki`(*filename=None*, *save_merged=False*)

Save as mediawiki to a file.

**filename: str**
If present, move the resulting file to this location.

**save_merged: bool**
If true, this will save the schema as a merged schema if it is a "withStandard" schema. If it is not a "withStandard" schema, this setting has no effect.

**Returns**
The newly created schema filename.

**Return type**
str

`save_as_xml`(*filename=None*, *save_merged=True*)

Save as XML to a file.

**filename: str**
If present, move the resulting file to this location.

**save_merged: bool**
If true, this will save the schema as a merged schema if it is a "withStandard" schema. If it is not a "withStandard" schema, this setting has no effect.

**Returns**
The name of the newly created schema file.

**Return type**
str

`schema_for_namespace`(*namespace*)

Return HedSchema object for this namespace.

**Parameters**
**namespace** (*str*) – The schema library name namespace.

**Returns**
The HED schema object for this schema.

**Return type**
*HedSchema*

`set_schema_prefix`(*schema_namespace*)

Set library namespace associated for this schema.

**Parameters**
**schema_namespace** (*str*) – Should be empty, or end with a colon.(Colon will be automated added if missing).

**property tags**

Return the tag schema section.

**Returns**
The tag section.

**Return type**
*HedSchemaTagSection*

**property unit_classes**

Return the unit classes schema section.

**Returns**
The unit classes section.

**Return type**
*HedSchemaUnitClassSection*

**property unit_modifiers**

Return the modifiers classes schema section

**Returns**
The unit modifiers section.

**Return type**
*HedSchemaSection*

**property valid_prefixes**

Return a list of all prefixes this schema will accept

**Returns**
A list of valid tag prefixes for this schema.

**Return type**
list

### Notes

- The return value is always length 1 if using a HedSchema.

**property value_classes**

Return the value classes schema section.

**Returns**
The value classes section.

**Return type**
*HedSchemaSection*

**property version**

The complete schema version, including prefix and library name(if applicable)

**property version_number**

The HED version of this schema.

**Returns**
The version of this schema.

**Return type**
str

**property with_standard**

The version of the base schema this is extended from, if it exists..

**Returns**
HED version or “”

**Return type**
str

### 3.3.3 hed.schema.hed_schema_base

Abstract base class for HedSchema and HedSchemaGroup, showing the common functionality

#### Classes

| | |
|---|---|
| *HedSchemaBase*() | Baseclass for schema and schema group. |

#### 3.3.3.1 hed.schema.hed_schema_base.HedSchemaBase

**class HedSchemaBase**

Bases: ABC

Baseclass for schema and schema group. Overriding the following functions will allow you to use the schema for validation etc.

**__init__**()

#### Methods

| | |
|---|---|
| *__init__*() | |
| *check_compliance*([check_for_warnings, name, ...]) | Check for HED3 compliance of this schema. |
| *find_tag_entry*(tag[, schema_namespace]) | Find the schema entry for a given source tag. |
| *get_formatted_version*() | The HED version string including namespace and library name if any of this schema. |
| *get_schema_versions*() | A list of HED version strings including namespace and library name if any of this schema. |
| *get_tag_entry*(name[, key_class, ...]) | Return the schema entry for this tag, if one exists. |
| *get_tags_with_attribute*(attribute[, key_class]) | Return tag entries with the given attribute. |
| *schema_for_namespace*(namespace) | Return the HedSchema for the library namespace. |

### Attributes

| | |
|---|---|
| *valid_prefixes* | Return a list of all prefixes this group will accept. |

**abstract check_compliance**(*check_for_warnings=True*, *name=None*, *error_handler=None*)

Check for HED3 compliance of this schema.

**Parameters**

- **check_for_warnings** (*bool*) – If True, checks for formatting issues like invalid characters, capitalization.
- **name** (*str*) – If present, use as the filename for context, rather than using the actual filename. Useful for temp filenames when supporting web services.
- **error_handler** (*ErrorHandler or None*) – Used to report errors. Uses a default one if none passed in.

**Returns**

A list of all warnings and errors found in the file. Each issue is a dictionary.

**Return type**

list

**abstract find_tag_entry**(*tag*, *schema_namespace=''*)

Find the schema entry for a given source tag.

**Parameters**

- **tag** (*str, HedTag*) – Any form of tag to look up. Can have an extension, value, etc.
- **schema_namespace** (*str*) – The schema namespace of the tag, if any.

**Returns**

The located tag entry for this tag. str: The remainder of the tag that isn't part of the base tag. list: A list of errors while converting.

**Return type**

*HedTagEntry*

### Notes

Works left to right (which is mostly relevant for errors).

**abstract get_formatted_version**()

The HED version string including namespace and library name if any of this schema.

**Returns**

The complete version of this schema including library name and namespace.

**Return type**

str

**abstract get_schema_versions**()

A list of HED version strings including namespace and library name if any of this schema.

**Returns**

The complete version of this schema including library name and namespace.

**Return type**
list

**abstract get_tag_entry**(*name*, *key_class=HedSectionKey.Tags*, *schema_namespace=''*)

Return the schema entry for this tag, if one exists.

**Parameters**

- **name** (*str*) – Any form of basic tag(or other section entry) to look up. This will not handle extensions or similar. If this is a tag, it can have a schema namespace, but it's not required
- **key_class** (HedSectionKey *or* *str*) – The type of entry to return.
- **schema_namespace** (*str*) – Only used on Tags. If incorrect, will return None.

**Returns**
The schema entry for the given tag.

**Return type**
*HedSchemaEntry*

**abstract get_tags_with_attribute**(*attribute*, *key_class=HedSectionKey.Tags*)

Return tag entries with the given attribute.

**Parameters**

- **attribute** (*str*) – A tag attribute. Eg HedKey.ExtensionAllowed
- **key_class** (HedSectionKey) – The HedSectionKey for the section to retrieve from.

**Returns**
A list of all tags with this attribute.

**Return type**
list

### Notes

- The result is cached so will be fast after first call.

**abstract schema_for_namespace**(*namespace*)

Return the HedSchema for the library namespace.

**Parameters**
**namespace** (*str*) – A schema library name namespace.

**Returns**
The specific schema for this library name namespace if exists.

**Return type**
*HedSchema* or None

**abstract property valid_prefixes**

Return a list of all prefixes this group will accept.

**Returns**
A list of strings representing valid prefixes for this group.

**Return type**
prefixes(list of str)

## 3.3.4 hed.schema.hed_schema_constants

### Classes

| | |
|---|---|
| *HedKey*() | Known property and attribute names. |
| *HedSectionKey*(value) | Kegs designating specific sections in a HedSchema object. |

#### 3.3.4.1 hed.schema.hed_schema_constants.HedKey

**class HedKey**

Bases: object

Known property and attribute names.

##### Notes

- These names should match the attribute values in the XML/wiki.

**__init__**()

##### Methods

| |
|---|
| *__init__*() |

### Attributes

| |
|---|
| AllowedCharacter |
| BoolProperty |
| DefaultUnits |
| DeprecatedFrom |
| ElementProperty |
| ExtensionAllowed |
| InLibrary |
| IsInheritedProperty |
| Recommended |
| RelatedTag |
| RequireChild |
| Required |
| Rooted |
| SIUnit |
| SIUnitModifier |
| SIUnitSymbolModifier |
| SuggestedTag |
| TagGroup |
| TakesValue |
| TopLevelTagGroup |
| Unique |
| UnitClass |
| UnitClassProperty |
| UnitModifierProperty |
| UnitPrefix |
| UnitProperty |
| UnitSymbol |
| ValueClass |
| ValueClassProperty |

#### 3.3.4.2 hed.schema.hed_schema_constants.HedSectionKey

**class HedSectionKey**(*value*)

Bases: Enum

Kegs designating specific sections in a HedSchema object.

**__init__**()

##### Attributes

| |
|---|
| Tags |
| UnitClasses |
| Units |
| UnitModifiers |
| ValueClasses |
| Attributes |
| Properties |

### 3.3.5 hed.schema.hed_schema_entry

#### Classes

| | |
|---|---|
| *HedSchemaEntry*(name, section) | A single node in a HedSchema. |
| *HedTagEntry*(*args, **kwargs) | A single tag entry in the HedSchema. |
| *UnitClassEntry*(*args, **kwargs) | A single unit class entry in the HedSchema. |
| *UnitEntry*(*args, **kwargs) | A single unit entry with modifiers in the HedSchema. |

#### 3.3.5.1 hed.schema.hed_schema_entry.HedSchemaEntry

**class HedSchemaEntry**(*name*, *section*)

Bases: object

A single node in a HedSchema.

The structure contains all the node information including attributes and properties.

**__init__**(*name*, *section*)

Constructor for HedSchemaEntry.

**Parameters**

- **name** (*str*) – The name of the entry.
- **section** (HedSchemaSection) – The section to which it belongs.

### Methods

| | |
|---|---|
| *__init__*(name, section) | Constructor for HedSchemaEntry. |
| *attribute_has_property*(attribute, property_name) | Return True if attribute has property. |
| *finalize_entry*(schema) | Called once after loading to set internal state. |
| get_known_attributes() | |
| *has_attribute*(attribute[, return_value]) | Checks for the existence of an attribute in this entry. |

### Attributes

| |
|---|
| section_key |

**attribute_has_property**(*attribute*, *property_name*)

Return True if attribute has property.

**Parameters**

- **attribute** (*str*) – Attribute name to check for property_name.
- **property_name** (*str*) – The property value to return.

**Returns**

Returns True if this entry has the property.

**Return type**

bool

**finalize_entry**(*schema*)

Called once after loading to set internal state.

**Parameters**

**schema** (HedSchema) – The schema that holds the rules.

**has_attribute**(*attribute*, *return_value=False*)

Checks for the existence of an attribute in this entry.

**Parameters**

- **attribute** (*str*) – The attribute to check for.
- **return_value** (*bool*) – If True, returns the actual value of the attribute. If False, returns a boolean indicating the presence of the attribute.

**Returns**

If return_value is False, returns True if the attribute exists and False otherwise. If return_value is True, returns the value of the attribute if it exists, else returns None.

**Return type**

bool or any

#### Notes

- The existence of an attribute does not guarantee its validity.

### 3.3.5.2 hed.schema.hed_schema_entry.HedTagEntry

**class HedTagEntry**(**args*, ***kwargs*)

Bases: *HedSchemaEntry*

A single tag entry in the HedSchema.

**__init__**(**args*, ***kwargs*)

Constructor for HedSchemaEntry.

**Parameters**

- **name** (*str*) – The name of the entry.
- **section** (HedSchemaSection) – The section to which it belongs.

#### Methods

| | |
|---|---|
| *__init__*(*args, **kwargs) | Constructor for HedSchemaEntry. |
| *attribute_has_property*(attribute, property_name) | Return True if attribute has property. |
| *base_tag_has_attribute*(tag_attribute) | Check if the base tag has a specific attribute. |
| *finalize_entry*(schema) | Called once after schema loading to set state. |
| get_known_attributes() | |
| *has_attribute*(attribute[, return_value]) | Returns th existence or value of an attribute in this entry. |

#### Attributes

| | |
|---|---|
| *parent* | Get the parent entry of this tag |
| *parent_name* | Gets the parent tag entry name |
| section_key | |

**attribute_has_property**(*attribute*, *property_name*)

Return True if attribute has property.

**Parameters**

- **attribute** (*str*) – Attribute name to check for property_name.
- **property_name** (*str*) – The property value to return.

**Returns**

Returns True if this entry has the property.

**Return type**

bool

**base_tag_has_attribute**(*tag_attribute*)

Check if the base tag has a specific attribute.

**Parameters**
**tag_attribute** (*str*) – A tag attribute.

**Returns**
True if the tag has the specified attribute. False, if otherwise.

**Return type**
bool

### Notes

This mostly is relevant for takes value tags.

**finalize_entry**(*schema*)

Called once after schema loading to set state.

**Parameters**
**schema** (HedSchema) – The schema that the rules come from.

**has_attribute**(*attribute*, *return_value=False*)

Returns th existence or value of an attribute in this entry.

This also checks parent tags for inheritable attributes like ExtensionAllowed.

**Parameters**

- **attribute** (*str*) – The attribute to check for.
- **return_value** (*bool*) – If True, returns the actual value of the attribute. If False, returns a boolean indicating the presence of the attribute.

**Returns**
If return_value is False, returns True if the attribute exists and False otherwise. If return_value is True, returns the value of the attribute if it exists, else returns None.

**Return type**
bool or any

### Notes

- The existence of an attribute does not guarantee its validity.

**property parent**

Get the parent entry of this tag

**property parent_name**

Gets the parent tag entry name

#### 3.3.5.3 hed.schema.hed_schema_entry.UnitClassEntry

**class UnitClassEntry**(**args*, ***kwargs*)

Bases: *HedSchemaEntry*

A single unit class entry in the HedSchema.

**__init__**(**args*, ***kwargs*)

Constructor for HedSchemaEntry.

**Parameters**

- **name** (*str*) – The name of the entry.
- **section** (HedSchemaSection) – The section to which it belongs.

### Methods

| | |
|---|---|
| *__init__*(*args, **kwargs) | Constructor for HedSchemaEntry. |
| *add_unit*(unit_entry) | Add the given unit entry to this unit class. |
| *attribute_has_property*(attribute, property_name) | Return True if attribute has property. |
| *finalize_entry*(schema) | Called once after schema load to set state. |
| get_known_attributes() | |
| *has_attribute*(attribute[, return_value]) | Checks for the existence of an attribute in this entry. |

### Attributes

| |
|---|
| section_key |

**add_unit**(*unit_entry*)

Add the given unit entry to this unit class.

**Parameters**
**unit_entry** (HedSchemaEntry) – Unit entry to add.

**attribute_has_property**(*attribute*, *property_name*)

Return True if attribute has property.

**Parameters**

- **attribute** (*str*) – Attribute name to check for property_name.
- **property_name** (*str*) – The property value to return.

**Returns**
Returns True if this entry has the property.

**Return type**
bool

**finalize_entry**(*schema*)

Called once after schema load to set state.

**Parameters**
**schema** (HedSchema) – The object with the schema rules.

**has_attribute**(*attribute*, *return_value=False*)

Checks for the existence of an attribute in this entry.

**Parameters**

- **attribute** (*str*) – The attribute to check for.
- **return_value** (*bool*) – If True, returns the actual value of the attribute. If False, returns a boolean indicating the presence of the attribute.

**Returns**
If return_value is False, returns True if the attribute exists and False otherwise. If return_value is True, returns the value of the attribute if it exists, else returns None.

**Return type**
bool or any

#### Notes

- The existence of an attribute does not guarantee its validity.

#### 3.3.5.4 hed.schema.hed_schema_entry.UnitEntry

**class UnitEntry**(**args*, ***kwargs*)

Bases: *HedSchemaEntry*

A single unit entry with modifiers in the HedSchema.

**__init__**(**args*, ***kwargs*)

Constructor for HedSchemaEntry.

**Parameters**

- **name** (*str*) – The name of the entry.
- **section** (HedSchemaSection) – The section to which it belongs.

#### Methods

| | |
|---|---|
| *__init__*(*args, **kwargs) | Constructor for HedSchemaEntry. |
| *attribute_has_property*(attribute, property_name) | Return True if attribute has property. |
| *finalize_entry*(schema) | Called once after loading to set internal state. |
| *get_conversion_factor*(unit_name) | Returns the conversion factor from combining this unit with the specified modifier |
| get_known_attributes() | |
| *has_attribute*(attribute[, return_value]) | Checks for the existence of an attribute in this entry. |

### Attributes

| |
|---|
| section_key |

**attribute_has_property**(*attribute*, *property_name*)

Return True if attribute has property.

**Parameters**

- **attribute** (*str*) – Attribute name to check for property_name.
- **property_name** (*str*) – The property value to return.

**Returns**
Returns True if this entry has the property.

**Return type**
bool

**finalize_entry**(*schema*)

Called once after loading to set internal state.

**Parameters**
**schema** (HedSchema) – The schema rules come from.

**get_conversion_factor**(*unit_name*)

Returns the conversion factor from combining this unit with the specified modifier

**Parameters**
**unit_name** (*str or None*) – the full name of the unit with modifier

**Returns**
Returns the conversion factor or None

**Return type**
conversion_factor(float or None)

**has_attribute**(*attribute*, *return_value=False*)

Checks for the existence of an attribute in this entry.

**Parameters**

- **attribute** (*str*) – The attribute to check for.
- **return_value** (*bool*) – If True, returns the actual value of the attribute. If False, returns a boolean indicating the presence of the attribute.

**Returns**
If return_value is False, returns True if the attribute exists and False otherwise. If return_value is True, returns the value of the attribute if it exists, else returns None.

**Return type**
bool or any

#### Notes

- The existence of an attribute does not guarantee its validity.

## 3.3.6 hed.schema.hed_schema_group

### Classes

| | |
|---|---|
| *HedSchemaGroup*(schema_list) | Container for multiple HedSchema objects. |

### 3.3.6.1 hed.schema.hed_schema_group.HedSchemaGroup

**class HedSchemaGroup**(*schema_list*)

Bases: *HedSchemaBase*

Container for multiple HedSchema objects.

#### Notes

- The container class is useful when library schema are included.
- You cannot save/load/etc the combined schema object directly.

**__init__**(*schema_list*)

Combine multiple HedSchema objects from a list.

**Parameters**
**schema_list** (*list*) – A list of HedSchema for the container.

**Returns**
the container created.

**Return type**
*HedSchemaGroup*

**Raises**
***HedFileError*** –

- Multiple schemas have the same library prefixes.
- Empty list passed

### Methods

| | |
|---|---|
| *__init__*(schema_list) | Combine multiple HedSchema objects from a list. |
| *check_compliance*([check_for_warnings, name, ...]) | Check for HED3 compliance of this schema. |
| *find_tag_entry*(tag[, schema_namespace]) | Find the schema entry for a given source tag. |
| *get_formatted_version*() | The HED version string including namespace and library name if any of this schema. |
| *get_schema_versions*() | A list of HED version strings including namespace and library name if any of this schema. |
| *get_tag_entry*(name[, key_class, ...]) | Return the schema entry for this tag, if one exists. |
| *get_tags_with_attribute*(attribute[, key_class]) | Return tag entries with the given attribute. |
| *schema_for_namespace*(namespace) | Return the HedSchema for the library namespace. |

### Attributes

| | |
|---|---|
| *valid_prefixes* | Return a list of all prefixes this group will accept. |

**check_compliance**(*check_for_warnings=True*, *name=None*, *error_handler=None*)

Check for HED3 compliance of this schema.

**Parameters**

- **check_for_warnings** (*bool*) – If True, checks for formatting issues like invalid characters, capitalization.
- **name** (*str*) – If present, use as the filename for context, rather than using the actual filename. Useful for temp filenames when supporting web services.
- **error_handler** (*ErrorHandler or None*) – Used to report errors. Uses a default one if none passed in.

**Returns**

A list of all warnings and errors found in the file. Each issue is a dictionary.

**Return type**

list

**find_tag_entry**(*tag*, *schema_namespace=''*)

Find the schema entry for a given source tag.

**Parameters**

- **tag** (*str, HedTag*) – Any form of tag to look up. Can have an extension, value, etc.
- **schema_namespace** (*str*) – The schema namespace of the tag, if any.

**Returns**

The located tag entry for this tag. str: The remainder of the tag that isn't part of the base tag. list: A list of errors while converting.

**Return type**

*HedTagEntry*

### Notes

Works left to right (which is mostly relevant for errors).

`get_formatted_version`()

The HED version string including namespace and library name if any of this schema.

**Returns**
The complete version of this schema including library name and namespace.

**Return type**
str

`get_schema_versions`()

A list of HED version strings including namespace and library name if any of this schema.

**Returns**
The complete version of this schema including library name and namespace.

**Return type**
list

`get_tag_entry`(*name*, *key_class=HedSectionKey.Tags*, *schema_namespace=''*)

Return the schema entry for this tag, if one exists.

**Parameters**

- **name** (*str*) – Any form of basic tag(or other section entry) to look up. This will not handle extensions or similar. If this is a tag, it can have a schema namespace, but it's not required
- **key_class** (HedSectionKey *or* *str*) – The type of entry to return.
- **schema_namespace** (*str*) – Only used on Tags. If incorrect, will return None.

**Returns**
The schema entry for the given tag.

**Return type**
*HedSchemaEntry*

`get_tags_with_attribute`(*attribute*, *key_class=HedSectionKey.Tags*)

Return tag entries with the given attribute.

**Parameters**

- **attribute** (*str*) – A tag attribute. Eg HedKey.ExtensionAllowed
- **key_class** (HedSectionKey) – The HedSectionKey for the section to retrieve from.

**Returns**
A list of all tags with this attribute.

**Return type**
list

#### Notes

- The result is cached so will be fast after first call.

**schema_for_namespace**(*namespace*)

Return the HedSchema for the library namespace.

**Parameters**
**namespace** (*str*) – A schema library name namespace.

**Returns**
The specific schema for this library name namespace if exists.

**Return type**
*HedSchema* or None

**property valid_prefixes**

Return a list of all prefixes this group will accept.

**Returns**
A list of strings representing valid prefixes for this group.

**Return type**
list

### 3.3.7 hed.schema.hed_schema_io

Utilities for loading and outputting HED schema.

#### Functions

| | |
|---|---|
| *from_string*(schema_string[, schema_format, ...]) | Create a schema from the given string. |
| *get_hed_xml_version*(xml_file_path) | Get the version number from a HED XML file. |
| *load_schema*([hed_path, schema_namespace]) | Load a schema from the given file or URL path. |
| *load_schema_version*([xml_version, xml_folder]) | Return a HedSchema or HedSchemaGroup extracted from xml_version field. |

#### 3.3.7.1 hed.schema.hed_schema_io.from_string

**from_string**(*schema_string*, *schema_format='.xml'*, *schema_namespace=None*)

Create a schema from the given string.

**Parameters**

- **schema_string** (*str*) – An XML or mediawiki file as a single long string.
- **schema_format** (*str*) – The schema format of the source schema string.
- **schema_namespace** (*str, None*) – The name_prefix all tags in this schema will accept.

**Returns**
The loaded schema.

**Return type**
(*HedSchema*)

**Raises**
*HedFileError* –

- If empty string or invalid extension is passed.
- Other fatal formatting issues with file

### Notes

- The loading is determined by file type.

#### 3.3.7.2 hed.schema.hed_schema_io.get_hed_xml_version

**get_hed_xml_version**(*xml_file_path*)

Get the version number from a HED XML file.

**Parameters**
**xml_file_path** (*str*) – The path to a HED XML file.

**Returns**
The version number of the HED XML file.

**Return type**
str

**Raises**
*HedFileError* –

- There is an issue loading the schema

#### 3.3.7.3 hed.schema.hed_schema_io.load_schema

**load_schema**(*hed_path=None*, *schema_namespace=None*)

Load a schema from the given file or URL path.

**Parameters**

- **hed_path** (*str or None*) – A filepath or url to open a schema from.
- **schema_namespace** (*str or None*) – The name_prefix all tags in this schema will accept.

**Returns**
The loaded schema.

**Return type**
*HedSchema*

**Raises**
*HedFileError* –

- Empty path passed
- Unknown extension
- Any fatal issues when loading the schema.

#### 3.3.7.4 hed.schema.hed_schema_io.load_schema_version

**load_schema_version**(*xml_version=None*, *xml_folder=None*)

Return a HedSchema or HedSchemaGroup extracted from xml_version field.

**Parameters**

- **xml_version** (*str or list or None*) – List or str specifying which official HED schemas to use. An empty string returns the latest version A json str format is also supported, based on the output of HedSchema.get_formatted_version
- **xml_folder** (*str*) – Path to a folder containing schema.

**Returns**

The schema or schema group extracted.

**Return type**

*HedSchema* or *HedSchemaGroup*

**Raises**

***HedFileError*** –

- The xml_version is not valid.
- A fatal error was encountered in parsing

### 3.3.8 hed.schema.hed_schema_section

#### Classes

| | |
|---|---|
| *HedSchemaSection*(section_key[, case_sensitive]) | Container with entries in one section of the schema. |
| *HedSchemaTagSection*(*args[, case_sensitive]) | A section of the schema. |
| *HedSchemaUnitClassSection*(section_key[, ...]) | |

#### 3.3.8.1 hed.schema.hed_schema_section.HedSchemaSection

**class HedSchemaSection**(*section_key*, *case_sensitive=True*)

Bases: `object`

Container with entries in one section of the schema.

**__init__**(*section_key*, *case_sensitive=True*)

Construct schema section.

**Parameters**

- **section_key** (HedSectionKey) – Name of the schema section.
- **case_sensitive** (*bool*) – If True, names are case-sensitive.

### Methods

| | |
|---|---|
| *__init__*(section_key[, case_sensitive]) | Construct schema section. |
| *get*(key) | Return the name associated with key. |
| *get_entries_with_attribute*(attribute_name[, ...]) | Return entries or names with given attribute. |
| *items*() | Return the items. |
| *keys*() | The names of the keys. |
| *values*() | All names of the sections. |

### Attributes

| |
|---|
| duplicate_names |
| section_key |

**get**(*key*)

Return the name associated with key.

**Parameters**
**key** (*str*) – The name of the key.

**get_entries_with_attribute**(*attribute_name*, *return_name_only=False*, *schema_namespace=''*)

Return entries or names with given attribute.

**Parameters**

- **attribute_name** (*str*) – The name of the attribute(generally a HedKey entry).
- **return_name_only** (*bool*) – If true, return the name as a string rather than the tag entry.
- **schema_namespace** (*str*) – Prepends given namespace to each name if returning names.

**Returns**
List of HedSchemaEntry or strings representing the names.

**Return type**
list

**items**()

Return the items.

**keys**()

The names of the keys.

**values**()

All names of the sections.

#### 3.3.8.2 hed.schema.hed_schema_section.HedSchemaTagSection

**class HedSchemaTagSection**(**args*, *case_sensitive=False*, ***kwargs*)

Bases: *HedSchemaSection*

A section of the schema.

**__init__**(**args*, *case_sensitive=False*, ***kwargs*)

Construct schema section.

**Parameters**

- **section_key** (*HedSectionKey*) – Name of the schema section.
- **case_sensitive** (*bool*) – If True, names are case-sensitive.

### Methods

| | |
|---|---|
| *__init__*(*args[, case_sensitive]) | Construct schema section. |
| *get*(key) | Return the name associated with key. |
| *get_entries_with_attribute*(attribute_name[, ...]) | Return entries or names with given attribute. |
| *items*() | Return the items. |
| *keys*() | The names of the keys. |
| *values*() | All names of the sections. |

### Attributes

| |
|---|
| duplicate_names |
| section_key |

**get**(*key*)

Return the name associated with key.

**Parameters**

**key** (*str*) – The name of the key.

**get_entries_with_attribute**(*attribute_name*, *return_name_only=False*, *schema_namespace=''*)

Return entries or names with given attribute.

**Parameters**

- **attribute_name** (*str*) – The name of the attribute(generally a HedKey entry).
- **return_name_only** (*bool*) – If true, return the name as a string rather than the tag entry.
- **schema_namespace** (*str*) – Prepends given namespace to each name if returning names.

**Returns**

List of HedSchemaEntry or strings representing the names.

**Return type**

list

**items**()

Return the items.

**keys**()

The names of the keys.

**values**()

All names of the sections.

#### 3.3.8.3 hed.schema.hed_schema_section.HedSchemaUnitClassSection

**class HedSchemaUnitClassSection**(*section_key*, *case_sensitive=True*)

Bases: *HedSchemaSection*

**__init__**(*section_key*, *case_sensitive=True*)

Construct schema section.

**Parameters**

- **section_key** (HedSectionKey) – Name of the schema section.
- **case_sensitive** (*bool*) – If True, names are case-sensitive.

### Methods

| | |
|---|---|
| *__init__*(section_key[, case_sensitive]) | Construct schema section. |
| *get*(key) | Return the name associated with key. |
| *get_entries_with_attribute*(attribute_name[, ...]) | Return entries or names with given attribute. |
| *items*() | Return the items. |
| *keys*() | The names of the keys. |
| *values*() | All names of the sections. |

### Attributes

| |
|---|
| duplicate_names |
| section_key |

**get**(*key*)

Return the name associated with key.

**Parameters**

**key** (*str*) – The name of the key.

**get_entries_with_attribute**(*attribute_name*, *return_name_only=False*, *schema_namespace=''*)

Return entries or names with given attribute.

**Parameters**

- **attribute_name** (*str*) – The name of the attribute(generally a HedKey entry).
- **return_name_only** (*bool*) – If true, return the name as a string rather than the tag entry.

- **schema_namespace** (*str*) – Prepends given namespace to each name if returning names.

**Returns**
List of HedSchemaEntry or strings representing the names.

**Return type**
list

**items**()
Return the items.

**keys**()
The names of the keys.

**values**()
All names of the sections.

### 3.3.9 hed.schema.schema_attribute_validators

The built-in functions to validate known attributes.

Template for the functions: attribute_checker_template(hed_schema, tag_entry, attribute_name, possible_values):

hed_schema (HedSchema): The schema to use for validation tag_entry (HedSchemaEntry): The schema entry for this tag. attribute_name (str): The name of this attribute

**returns**
bool

#### Functions

| | |
|---|---|
| *tag_exists_base_schema_check*(hed_schema, ...) | Check if the single tag is a partnered schema tag |
| *tag_exists_check*(hed_schema, tag_entry, ...) | Check if the list of possible tags exists in the schema. |
| *tag_is_deprecated_check*(hed_schema, ...) | Check if the tag has a valid deprecatedFrom attribute, and that any children have it |
| *tag_is_placeholder_check*(hed_schema, ...) | Check if comma separated list has valid HedTags. |
| *unit_class_exists*(hed_schema, tag_entry, ...) | |
| *unit_exists*(hed_schema, tag_entry, ...) | |
| *value_class_exists*(hed_schema, tag_entry, ...) | |

#### 3.3.9.1 hed.schema.schema_attribute_validators.tag_exists_base_schema_check

**tag_exists_base_schema_check**(*hed_schema*, *tag_entry*, *attribute_name*)
Check if the single tag is a partnered schema tag

**Parameters**

- **hed_schema** (HedSchema) – The schema to use for validation
- **tag_entry** (HedSchemaEntry) – The schema entry for this tag.
- **attribute_name** (*str*) – The name of this attribute

**Returns**
A list of issues. Each issue is a dictionary.

**Return type**
list

#### 3.3.9.2 hed.schema.schema_attribute_validators.tag_exists_check

tag_exists_check(*hed_schema*, *tag_entry*, *attribute_name*)

Check if the list of possible tags exists in the schema.

**Parameters**

- **hed_schema** (HedSchema) – The schema to use for validation
- **tag_entry** (HedSchemaEntry) – The schema entry for this tag.
- **attribute_name** (*str*) – The name of this attribute

**Returns**
A list of issues. Each issue is a dictionary.

**Return type**
list

#### 3.3.9.3 hed.schema.schema_attribute_validators.tag_is_deprecated_check

tag_is_deprecated_check(*hed_schema*, *tag_entry*, *attribute_name*)

Check if the tag has a valid deprecatedFrom attribute, and that any children have it

**Parameters**

- **hed_schema** (HedSchema) – The schema to use for validation
- **tag_entry** (HedSchemaEntry) – The schema entry for this tag.
- **attribute_name** (*str*) – The name of this attribute

**Returns**
A list of issues. Each issue is a dictionary.

**Return type**
list

#### 3.3.9.4 hed.schema.schema_attribute_validators.tag_is_placeholder_check

tag_is_placeholder_check(*hed_schema*, *tag_entry*, *attribute_name*)

Check if comma separated list has valid HedTags.

**Parameters**

- **hed_schema** (HedSchema) – The schema to use for validation
- **tag_entry** (HedSchemaEntry) – The schema entry for this tag.
- **attribute_name** (*str*) – The name of this attribute

**Returns**
A list of issues. Each issue is a dictionary.

**Return type**
list

#### 3.3.9.5 hed.schema.schema_attribute_validators.unit_class_exists

**unit_class_exists**(*hed_schema*, *tag_entry*, *attribute_name*)

#### 3.3.9.6 hed.schema.schema_attribute_validators.unit_exists

**unit_exists**(*hed_schema*, *tag_entry*, *attribute_name*)

#### 3.3.9.7 hed.schema.schema_attribute_validators.value_class_exists

**value_class_exists**(*hed_schema*, *tag_entry*, *attribute_name*)

### 3.3.10 hed.schema.schema_compare

#### Functions

| | |
|---|---|
| *compare_differences*(schema1, schema2[, ...]) | Compare the tags in two schemas, this finds any differences |
| *compare_schemas*(schema1, schema2[, ...]) | Compare two schemas section by section. |
| *find_matching_tags*(schema1, schema2[, ...]) | Compare the tags in two library schemas. |

#### 3.3.10.1 hed.schema.schema_compare.compare_differences

**compare_differences**(*schema1*, *schema2*, *output='raw'*, *attribute_filter=None*, *sections=(<HedSectionKey.Tags: 'tags'>, )*, *include_summary=True*)

Compare the tags in two schemas, this finds any differences

**Parameters**

- **schema1** (HedSchema) – The first schema to be compared.
- **schema2** (HedSchema) – The second schema to be compared.
- **output** (*str*) – 'raw' (default) returns a tuple of python object dicts with raw results. 'string' returns a single string 'dict' returns a json-style python dictionary that can be converted to JSON
- **attribute_filter** (*str, optional*) – The attribute to filter entries by. Entries without this attribute are skipped. The most common use would be HedKey.InLibrary If it evaluates to False, no filtering is performed.
- **sections** (*list or None*) – the list of sections to compare. By default, just the tags section. If None, checks all sections including header, prologue, and epilogue.
- **include_summary** (*bool*) – If True, adds the 'summary' dict to the dict return option, and prints it with the string option. Lists the names of all the nodes that are missing or different.

**Returns**

- Tuple with dict entries (not_in_schema1, not_in_schema1, unequal_entries).

- Formatted string with the output ready for printing.
- A Python dictionary with the output ready to be converted to JSON (for web output).

**Return type**
tuple, str or dict

**Notes: The underlying dictionaries are:**

- not_in_schema1(dict): Entries present in schema2 but not in schema1.
- not_in_schema2(dict): Entries present in schema1 but not in schema2.
- unequal_entries(dict): Entries that differ between the two schemas.

#### 3.3.10.2 hed.schema.schema_compare.compare_schemas

**compare_schemas**(*schema1*, *schema2*, *attribute_filter='inLibrary'*, *sections=(<HedSectionKey.Tags: 'tags'>, )*)

Compare two schemas section by section. The function records matching entries, entries present in one schema but not in the other, and unequal entries.

**Parameters**

- **schema1** (HedSchema) – The first schema to be compared.
- **schema2** (HedSchema) – The second schema to be compared.
- **attribute_filter** (*str, optional*) – The attribute to filter entries by. Entries without this attribute are skipped. The most common use would be HedKey.InLibrary If it evaluates to False, no filtering is performed.
- **sections** (*list*) – the list of sections to compare. By default, just the tags section. If None, checks all sections including header, prologue, and epilogue.

Returns: tuple: A tuple containing four dictionaries:

- matches(dict): Entries present in both schemas and are equal.
- not_in_schema1(dict): Entries present in schema2 but not in schema1.
- not_in_schema2(dict): Entries present in schema1 but not in schema2.
- unequal_entries(dict): Entries present in both schemas but are not equal.

#### 3.3.10.3 hed.schema.schema_compare.find_matching_tags

**find_matching_tags**(*schema1*, *schema2*, *output='raw'*, *sections=(<HedSectionKey.Tags: 'tags'>, )*, *include_summary=True*)

Compare the tags in two library schemas. This finds tags with the same term.

**Parameters**

- **schema1** (HedSchema) – The first schema to be compared.
- **schema2** (HedSchema) – The second schema to be compared.
- **output** (*str*) – Defaults to returning a python object dicts. 'string' returns a single string 'dict' returns a json style dictionary
- **sections** (*list*) – the list of sections to compare. By default, just the tags section. If None, checks all sections including header, prologue, and epilogue.

- **include_summary** (*bool*) – If True, adds the 'summary' dict to the dict return option, and prints it with the string option. Lists the names of all the nodes that are missing or different.

**Returns**
A dictionary containing matching entries in the Tags section of both schemas.

**Return type**
dict, json style dict, or str

### 3.3.11 hed.schema.schema_compliance

Utilities for HED schema checking.

#### Functions

| | |
|---|---|
| *check_compliance*(hed_schema[, ...]) | Check for hed3 compliance of a schema object. |

#### 3.3.11.1 hed.schema.schema_compliance.check_compliance

**check_compliance**(*hed_schema*, *check_for_warnings=True*, *name=None*, *error_handler=None*)

Check for hed3 compliance of a schema object.

**Parameters**

- **hed_schema** (HedSchema) – HedSchema object to check for hed3 compliance.
- **check_for_warnings** (*bool*) – If True, check for formatting issues like invalid characters, capitalization, etc.
- **name** (*str*) – If present, will use as filename for context.
- **error_handler** (ErrorHandler *or None*) – Used to report errors. Uses a default one if none passed in.

**Returns**
A list of all warnings and errors found in the file. Each issue is a dictionary.

**Return type**
list

**Raises**
**ValueError** –

- Trying to validate a HedSchemaGroup directly

#### Classes

| | |
|---|---|
| *SchemaValidator*(hed_schema[, ...]) | Validator class to wrap some code. |

#### 3.3.11.2 hed.schema.schema_compliance.SchemaValidator

**class SchemaValidator**(*hed_schema*, *check_for_warnings=True*, *error_handler=None*)

Bases: object

Validator class to wrap some code. In general, just call check_compliance.

**__init__**(*hed_schema*, *check_for_warnings=True*, *error_handler=None*)

##### Methods

| | |
|---|---|
| *__init__*(hed_schema[, check_for_warnings, ...]) | |
| *check_attributes*() | Returns issues from validating known attributes in all sections |
| *check_duplicate_names*() | Return issues for any duplicate names in all sections. |
| *check_invalid_chars*() | Returns issues for bad chars in terms or descriptions. |
| *check_unknown_attributes*() | Returns issues for any unknown attributes in any section |

##### Attributes

| |
|---|
| attribute_validators |

**check_attributes**()

Returns issues from validating known attributes in all sections

**check_duplicate_names**()

Return issues for any duplicate names in all sections.

**check_invalid_chars**()

Returns issues for bad chars in terms or descriptions.

**check_unknown_attributes**()

Returns issues for any unknown attributes in any section

### 3.3.12 hed.schema.schema_io

### Modules

| | |
|---|---|
| *hed.schema.schema_io.base2schema* | |
| *hed.schema.schema_io.schema2base* | Baseclass for mediawiki/xml writers |
| *hed.schema.schema_io.schema2wiki* | Allows output of HedSchema objects as .mediawiki format |
| *hed.schema.schema_io.schema2xml* | Allows output of HedSchema objects as .xml format |
| *hed.schema.schema_io.schema_util* | Utilities for writing content to files and for other file manipulation. |
| *hed.schema.schema_io.wiki2schema* | This module is used to create a HedSchema object from a .mediawiki file. |
| *hed.schema.schema_io.wiki_constants* | |
| *hed.schema.schema_io.xml2schema* | This module is used to create a HedSchema object from an XML file or tree. |
| *hed.schema.schema_io.xml_constants* | |

#### 3.3.12.1 hed.schema.schema_io.base2schema

### Classes

| | |
|---|---|
| *SchemaLoader*(filename[, schema_as_string]) | Baseclass for schema loading, to handle basic errors and partnered schemas |

### hed.schema.schema_io.base2schema.SchemaLoader

**class SchemaLoader**(*filename*, *schema_as_string=None*)

Bases: ABC

Baseclass for schema loading, to handle basic errors and partnered schemas

Expected usage is SchemaLoaderXML.load(filename)

SchemaLoaderXML(filename) will load just the header_attributes

**__init__**(*filename*, *schema_as_string=None*)

Loads the given schema from one of the two parameters.

**Parameters**

- **filename** (*str or None*) – A valid filepath or None
- **schema_as_string** (*str or None*) – A full schema as text or None

#### Methods

| | |
|---|---|
| *__init__*(filename[, schema_as_string]) | Loads the given schema from one of the two parameters. |
| *load*([filename, schema_as_string]) | Loads and returns the schema, including partnered schema if applicable. |

#### Attributes

| | |
|---|---|
| *schema* | The partially loaded schema if you are after just header attributes. |

**classmethod load**(*filename=None*, *schema_as_string=None*)

Loads and returns the schema, including partnered schema if applicable.

**Parameters**

- **filename** (*str or None*) – A valid filepath or None
- **schema_as_string** (*str or None*) – A full schema as text or None

**Returns**
The new schema

**Return type**
schema(*HedSchema*)

**property schema**

The partially loaded schema if you are after just header attributes.

### 3.3.12.2 hed.schema.schema_io.schema2base

Baseclass for mediawiki/xml writers

#### Classes

| | |
|---|---|
| *Schema2Base*() | |

#### hed.schema.schema_io.schema2base.Schema2Base

**class Schema2Base**

Bases: object

**__init__**()

### Methods

| | |
|---|---|
| *__init__*() | |
| *process_schema*(hed_schema[, save_merged]) | Takes a HedSchema object and returns a list of strings representing its .mediawiki version. |

**classmethod process_schema**(*hed_schema*, *save_merged=False*)

Takes a HedSchema object and returns a list of strings representing its .mediawiki version.

**Parameters**

- **hed_schema** (HedSchema) –
- **save_merged** (*bool*) – If True, this will save the schema as a merged schema if it is a "withStandard" schema. If it is not a "withStandard" schema, this setting has no effect.

**Returns**

**converted_output** – Varies based on inherited class

**Return type**

Any

#### 3.3.12.3 hed.schema.schema_io.schema2wiki

Allows output of HedSchema objects as .mediawiki format

### Classes

| | |
|---|---|
| *Schema2Wiki*() | |

### hed.schema.schema_io.schema2wiki.Schema2Wiki

**class Schema2Wiki**

Bases: *Schema2Base*

**__init__**()

### Methods

| | |
|---|---|
| *__init__*() | |
| *process_schema*(hed_schema[, save_merged]) | Takes a HedSchema object and returns a list of strings representing its .mediawiki version. |

**classmethod process_schema**(*hed_schema*, *save_merged=False*)

Takes a HedSchema object and returns a list of strings representing its .mediawiki version.

**Parameters**

- **hed_schema** (HedSchema) –
- **save_merged** (*bool*) – If True, this will save the schema as a merged schema if it is a "withStandard" schema. If it is not a "withStandard" schema, this setting has no effect.

**Returns**
**converted_output** – Varies based on inherited class

**Return type**
Any

#### 3.3.12.4 hed.schema.schema_io.schema2xml

Allows output of HedSchema objects as .xml format

### Classes

| |
|---|
| *Schema2XML*() |

### hed.schema.schema_io.schema2xml.Schema2XML

**class Schema2XML**

Bases: *Schema2Base*

**__init__**()

### Methods

| |
|---|
| *__init__*() |

| | |
|---|---|
| *process_schema*(hed_schema[, save_merged]) | Takes a HedSchema object and returns a list of strings representing its .mediawiki version. |

**classmethod process_schema**(*hed_schema*, *save_merged=False*)

Takes a HedSchema object and returns a list of strings representing its .mediawiki version.

**Parameters**

- **hed_schema** (HedSchema) –
- **save_merged** (*bool*) – If True, this will save the schema as a merged schema if it is a "withStandard" schema. If it is not a "withStandard" schema, this setting has no effect.

**Returns**
**converted_output** – Varies based on inherited class

**Return type**
Any

#### 3.3.12.5 hed.schema.schema_io.schema_util

Utilities for writing content to files and for other file manipulation.

### Functions

| | |
|---|---|
| *get_api_key*() | Tries to get the GitHub access token from the environment. Defaults to above value if not found. |
| *make_url_request*(resource_url[, ...]) | Make a request and adds the above GitHub access credentials. |
| *move_file*(input_path, target_path) | If target_path is not empty, move input file to target file |
| *url_to_file*(resource_url) | Write data from a URL resource into a file. |
| *url_to_string*(resource_url) | Get the data from the specified url as a string. |
| *write_strings_to_file*(output_strings[, ...]) | Write output strings to a temporary file. |
| *write_xml_tree_2_xml_file*(xml_tree[, extension]) | Write an XML element tree object into an XML file. |

### hed.schema.schema_io.schema_util.get_api_key

**get_api_key**()

Tries to get the GitHub access token from the environment. Defaults to above value if not found.

**Returns**
A GitHub access key or an empty string.

### hed.schema.schema_io.schema_util.make_url_request

**make_url_request**(*resource_url*, *try_authenticate=True*)

Make a request and adds the above GitHub access credentials.

**Parameters**

- **resource_url** (*str*) – The url to retrieve.
- **try_authenticate** (*bool*) – If true add the above credentials.

**Returns**
url_request

### hed.schema.schema_io.schema_util.move_file

**move_file**(*input_path*, *target_path*)

If target_path is not empty, move input file to target file

**Parameters**

- **input_path** (*str*) – Path to an existing file
- **target_path** (*str or None*) – Path to move this file to If None, the function does nothing and returns input_path

**Returns**
the original or moved filepath

**Return type**
filepath(str)

### hed.schema.schema_io.schema_util.url_to_file

`url_to_file`(*resource_url*)

Write data from a URL resource into a file. Data is decoded as unicode.

**Parameters**
**resource_url** (*str*) – The URL to the resource.

**Returns**
The local temporary filename for the downloaded file,

**Return type**
str

### hed.schema.schema_io.schema_util.url_to_string

`url_to_string`(*resource_url*)

Get the data from the specified url as a string.

**Parameters**
**resource_url** (*str*) – The URL to the resource.

**Returns**
The data at the target url.

**Return type**
str

### hed.schema.schema_io.schema_util.write_strings_to_file

`write_strings_to_file`(*output_strings*, *extension=None*)

Write output strings to a temporary file.

**Parameters**

- **output_strings** (*[str], str*) – Strings to output one per line.
- **extension** (*str*) – File extension of the temporary file.

**Returns**
Opened temporary file.

**Return type**
file

#### hed.schema.schema_io.schema_util.write_xml_tree_2_xml_file

**write_xml_tree_2_xml_file**(*xml_tree*, *extension='.xml'*)

Write an XML element tree object into an XML file.

**Parameters**

- **xml_tree** (*Element*) – An element representing an XML file.
- **extension** (*string*) – The file extension to use for the temporary file.

**Returns**

Name of the temporary file.

**Return type**

str

### 3.3.12.6 hed.schema.schema_io.wiki2schema

This module is used to create a HedSchema object from a .mediawiki file.

#### Classes

| | |
|---|---|
| *SchemaLoaderWiki*(filename[, schema_as_string]) | Loads MediaWiki schemas from filenames or strings. |

#### hed.schema.schema_io.wiki2schema.SchemaLoaderWiki

**class SchemaLoaderWiki**(*filename*, *schema_as_string=None*)

Bases: *SchemaLoader*

Loads MediaWiki schemas from filenames or strings.

Expected usage is SchemaLoaderWiki.load(filename)

SchemaLoaderWiki(filename) will load just the header_attributes

**__init__**(*filename*, *schema_as_string=None*)

Loads the given schema from one of the two parameters.

**Parameters**

- **filename** (*str or None*) – A valid filepath or None
- **schema_as_string** (*str or None*) – A full schema as text or None

##### Methods

| | |
|---|---|
| *__init__*(filename[, schema_as_string]) | Loads the given schema from one of the two parameters. |
| *load*([filename, schema_as_string]) | Loads and returns the schema, including partnered schema if applicable. |

### Attributes

| | |
|---|---|
| *schema* | The partially loaded schema if you are after just header attributes. |

**classmethod load**(*filename=None*, *schema_as_string=None*)

Loads and returns the schema, including partnered schema if applicable.

**Parameters**

- **filename** (*str or None*) – A valid filepath or None
- **schema_as_string** (*str or None*) – A full schema as text or None

**Returns**
The new schema

**Return type**
schema(*HedSchema*)

**property schema**

The partially loaded schema if you are after just header attributes.

#### 3.3.12.7 hed.schema.schema_io.wiki_constants

### Classes

| | |
|---|---|
| *HedWikiSection*() | |

#### hed.schema.schema_io.wiki_constants.HedWikiSection

**class HedWikiSection**

Bases: object

**__init__**()

### Methods

| | |
|---|---|
| *__init__*() | |

### Attributes

| Attributes |
|---|
| EndHed |
| EndSchema |
| Epilogue |
| HeaderLine |
| Prologue |
| Properties |
| Schema |
| UnitModifiers |
| UnitsClasses |
| ValueClasses |

#### 3.3.12.8 hed.schema.schema_io.xml2schema

This module is used to create a HedSchema object from an XML file or tree.

### Classes

| | |
|---|---|
| *SchemaLoaderXML*(filename[, schema_as_string]) | Loads XML schemas from filenames or strings. |

### hed.schema.schema_io.xml2schema.SchemaLoaderXML

**class SchemaLoaderXML**(*filename*, *schema_as_string=None*)

Bases: *SchemaLoader*

Loads XML schemas from filenames or strings.

Expected usage is SchemaLoaderXML.load(filename)

SchemaLoaderXML(filename) will load just the header_attributes

**__init__**(*filename*, *schema_as_string=None*)

Loads the given schema from one of the two parameters.

**Parameters**

- **filename** (*str or None*) – A valid filepath or None
- **schema_as_string** (*str or None*) – A full schema as text or None

#### Methods

| | |
|---|---|
| *__init__*(filename[, schema_as_string]) | Loads the given schema from one of the two parameters. |
| *load*([filename, schema_as_string]) | Loads and returns the schema, including partnered schema if applicable. |

#### Attributes

| | |
|---|---|
| *schema* | The partially loaded schema if you are after just header attributes. |

**classmethod load**(*filename=None*, *schema_as_string=None*)

Loads and returns the schema, including partnered schema if applicable.

**Parameters**

- **filename** (*str or None*) – A valid filepath or None
- **schema_as_string** (*str or None*) – A full schema as text or None

**Returns**
The new schema

**Return type**
schema(*HedSchema*)

**property schema**

The partially loaded schema if you are after just header attributes.

#### 3.3.12.9 hed.schema.schema_io.xml_constants

### 3.3.13 hed.schema.schema_validation_util

Utilities used in HED validation/loading using a HED schema.

#### Functions

| | |
|---|---|
| *find_rooted_entry*(tag_entry, schema, ...) | This semi-validates rooted tags, raising an exception on major errors |
| *validate_attributes*(attrib_dict, filename) | Validate attributes in the dictionary. |
| *validate_library_name*(library_name) | Check the validity of the library name. |
| *validate_present_attributes*(attrib_dict, ...) | Validate combinations of attributes |
| *validate_schema_description*(tag_name, ...) | Check the description of a single schema term. |
| *validate_schema_term*(hed_term) | Check short tag for capitalization and illegal characters. |
| *validate_version_string*(version_string) | Check validity of the version. |

#### 3.3.13.1 hed.schema.schema_validation_util.find_rooted_entry

**find_rooted_entry**(*tag_entry*, *schema*, *loading_merged*)

This semi-validates rooted tags, raising an exception on major errors

**Parameters**

- **tag_entry** (HedTagEntry) – the possibly rooted tag
- **schema** (HedSchema) – The schema being loaded
- **loading_merged** (*bool*) – If this schema was already merged before loading

**Returns**

**The base tag entry from the standard schema**
Returns None if this tag isn't rooted

**Return type**
rooted_tag(*HedTagEntry* or None)

**Raises**
*HedFileError* –

- A rooted attribute is found in a non-paired schema
- A rooted attribute is not a string
- A rooted attribute was found on a non-root node in an unmerged schema.
- A rooted attribute is found on a root node in a merged schema.
- A rooted attribute indicates a tag that doesn't exist in the base schema.

#### 3.3.13.2 hed.schema.schema_validation_util.validate_attributes

**validate_attributes**(*attrib_dict*, *filename*)

Validate attributes in the dictionary.

**Parameters**

- **attrib_dict** (*dict*) – Dictionary of attributes to be evaluated.
- **filename** (*str*) – File name to use in reporting errors.

**Returns**
List of issues. Each issue is a dictionary.

**Return type**
list

**Raises**
*HedFileError* –

- Invalid library name
- Version not present
- Invalid combinations of attributes in header

#### 3.3.13.3 hed.schema.schema_validation_util.validate_library_name

validate_library_name(*library_name*)

Check the validity of the library name.

**Parameters**
**library_name** (*str*) – Name of the library.

**Returns**
If not False, string indicates the issue.

**Return type**
bool or str

#### 3.3.13.4 hed.schema.schema_validation_util.validate_present_attributes

validate_present_attributes(*attrib_dict*, *filename*)

Validate combinations of attributes

**Parameters**

- **attrib_dict** (*dict*) – Dictionary of attributes to be evaluated.
- **filename** (*str*) – File name to use in reporting errors.

**Returns**
List of issues. Each issue is a dictionary.

**Return type**
list

**Raises**
*HedFileError* –

- withStandard is found in th header, but a library attribute is not specified

#### 3.3.13.5 hed.schema.schema_validation_util.validate_schema_description

validate_schema_description(*tag_name*, *hed_description*)

Check the description of a single schema term.

**Parameters**

- **tag_name** (*str*) – A single hed tag - not validated here, just used for error messages.
- **hed_description** (*str*) – The description string to validate.

**Returns**
A list of all formatting issues found in the description.

**Return type**
list

#### 3.3.13.6 hed.schema.schema_validation_util.validate_schema_term

validate_schema_term(*hed_term*)

Check short tag for capitalization and illegal characters.

**Parameters**
**hed_term** (*str*) – A single hed term.

**Returns**
A list of all formatting issues found in the term. Each issue is a dictionary.

**Return type**
list

#### 3.3.13.7 hed.schema.schema_validation_util.validate_version_string

validate_version_string(*version_string*)

Check validity of the version.

**Parameters**
**version_string** (*str*) – A version string.

**Returns**
If not False, string indicates the issue.

**Return type**
bool or str

## 3.4 hed.tools

HED remodeling, analysis and summarization tools.

### Modules

| | |
|---|---|
| *hed.tools.analysis* | Basic analysis tools. |
| *hed.tools.bids* | Models for BIDS datasets and files. |
| *hed.tools.remodeling* | Remodeling tools for revising and summarizing tabular files. |
| *hed.tools.util* | Data and file handling utilities. |
| *hed.tools.visualization* | |

### 3.4.1 hed.tools.analysis

Basic analysis tools.

### Modules

| | |
|---|---|
| hed.tools.analysis.analysis_util | Utilities for assembly, analysis, and searching. |
| hed.tools.analysis.annotation_util | Utilities to facilitate annotation of events in BIDS. |
| hed.tools.analysis.column_name_summary | Summarizes the unique column names in a dataset. |
| hed.tools.analysis.event_manager | Manages events of temporal extent. |
| hed.tools.analysis.file_dictionary | Representation of a file dictionary keyed by entity indices. |
| hed.tools.analysis.hed_tag_counts | Counts of HED tags in a file's annotations. |
| hed.tools.analysis.hed_tag_manager | Manager for the HED tags in a tabular file. |
| hed.tools.analysis.hed_type | Manages a type variable and its associated context. |
| hed.tools.analysis.hed_type_counts | Manages the counts of tags such as Condition-variable and task. |
| hed.tools.analysis.hed_type_defs | Manages definitions associated with a type such as condition-variable. |
| hed.tools.analysis.hed_type_factors | Manages factor information for a tabular file. |
| hed.tools.analysis.hed_type_manager | Manager for type factors and type definitions. |
| hed.tools.analysis.key_map | A map of column value keys into new column values. |
| hed.tools.analysis.tabular_summary | Summarize the contents of tabular files. |
| hed.tools.analysis.temporal_event | |

#### 3.4.1.1 hed.tools.analysis.analysis_util

Utilities for assembly, analysis, and searching.

### Functions

| | |
|---|---|
| assemble_hed(data_input, sidecar, schema[, ...]) | Return assembled HED annotations in a dataframe. |
| get_expression_parsers(queries[, query_names]) | Returns a list of expression parsers and query_names. |
| hed_to_str(contents[, remove_parentheses]) | |
| search_strings(hed_strings, queries[, ...]) | Returns a DataFrame of factors based on results of queries. |

### hed.tools.analysis.analysis_util.assemble_hed

**assemble_hed**(*data_input*, *sidecar*, *schema*, *columns_included=None*, *expand_defs=False*)

Return assembled HED annotations in a dataframe.

**Parameters**

- **data_input** (TabularInput) – The tabular input file whose HED annotations are to be assembled.
- **sidecar** (Sidecar) – Sidecar with definitions.
- **schema** (HedSchema) – Hed schema
- **columns_included** (*list or None*) – A list of additional column names to include. If None, only the list of assembled tags is included.

- **expand_defs** (*bool*) – If True, definitions are expanded when the events are assembled.

**Returns**
A DataFrame with the assembled events. dict: A dictionary with definition names as keys and definition content strings as values.

**Return type**
DataFrame or None

### hed.tools.analysis.analysis_util.get_expression_parsers

**get_expression_parsers**(*queries*, *query_names=None*)

Returns a list of expression parsers and query_names.

**Parameters**

- **queries** (*list*) – A list of query strings or QueryParser objects
- **query_names** (*list*) – A list of column names for results of queries. If missing — query_1, query_2, etc.

**Returns**
DataFrame - containing the search strings

**Raises**
**ValueError** –

- If query names are invalid or duplicated.

### hed.tools.analysis.analysis_util.hed_to_str

**hed_to_str**(*contents*, *remove_parentheses=False*)

### hed.tools.analysis.analysis_util.search_strings

**search_strings**(*hed_strings*, *queries*, *query_names=None*)

Returns a DataFrame of factors based on results of queries.

**Parameters**

- **hed_strings** (*list*) – A list of HedString objects (empty entries or None entries are 0's)
- **queries** (*list*) – A list of query strings or QueryParser objects
- **query_names** (*list*) – A list of column names for results of queries. If missing — query_1, query_2, etc.

**Returns**
DataFrame - containing the factor vectors with results of the queries

**Raises**
**ValueError** –

- If query names are invalid or duplicated.

#### 3.4.1.2 hed.tools.analysis.annotation_util

Utilities to facilitate annotation of events in BIDS.

### Functions

| | |
|---|---|
| *check_df_columns*(df[, required_cols]) | Return a list of the specified columns that are missing from a dataframe. |
| *df_to_hed*(dataframe[, description_tag]) | Create sidecar-like dictionary from a 4-column dataframe. |
| *extract_tags*(hed_string, search_tag) | Extract all instances of specified tag from a tag_string. |
| *generate_sidecar_entry*(column_name[, ...]) | Create a sidecar column dictionary for column. |
| *hed_to_df*(sidecar_dict[, col_names]) | Return a 4-column dataframe of HED portions of sidecar. |
| *merge_hed_dict*(sidecar_dict, hed_dict) | Update a JSON sidecar based on the hed_dict values. |
| *trim_back*(tag_string) | Return a trimmed copy of tag_string. |
| *trim_front*(tag_string) | Return a copy of tag_string with leading blanks and commas removed. |

### hed.tools.analysis.annotation_util.check_df_columns

**check_df_columns**(*df*, *required_cols=('column_name', 'column_value', 'description', 'HED')*)

Return a list of the specified columns that are missing from a dataframe.

**Parameters**

- **df** (*DataFrame*) – Spreadsheet to check the columns of.
- **required_cols** (*tuple*) – List of column names that must be present.

**Returns**

List of column names that are missing.

**Return type**

list

### hed.tools.analysis.annotation_util.df_to_hed

**df_to_hed**(*dataframe*, *description_tag=True*)

Create sidecar-like dictionary from a 4-column dataframe.

**Parameters**

- **dataframe** (*DataFrame*) – A four-column Pandas DataFrame with specific columns.
- **description_tag** (*bool*) – If True description tag is included.

**Returns**

A dictionary compatible with BIDS JSON tabular file that includes HED.

**Return type**

dict

#### Notes

- The DataFrame must have the columns with names: column_name, column_value, description, and HED.

### hed.tools.analysis.annotation_util.extract_tags

**extract_tags**(*hed_string*, *search_tag*)

Extract all instances of specified tag from a tag_string.

**Parameters**

- **hed_string** (*str*) – Tag string from which to extract tag.
- **search_tag** (*str*) – HED tag to extract.

**Returns**

- str: Tag string without the tags.
- list: A list of the tags that were extracted, for example descriptions.

**Return type**
tuple

### hed.tools.analysis.annotation_util.generate_sidecar_entry

**generate_sidecar_entry**(*column_name*, *column_values=None*)

Create a sidecar column dictionary for column.

**Parameters**

- **column_name** (*str*) – Name of the column.
- **column_values** – List of column values.

### hed.tools.analysis.annotation_util.hed_to_df

**hed_to_df**(*sidecar_dict*, *col_names=None*)

Return a 4-column dataframe of HED portions of sidecar.

**Parameters**

- **sidecar_dict** (*dict*) – A dictionary conforming to BIDS JSON events sidecar format.
- **col_names** (*list, None*) – A list of the cols to include in the flattened side car.

**Returns**
Four-column spreadsheet representing HED portion of sidecar.

**Return type**
DataFrame

#### Notes

- The returned DataFrame has columns: column_name, column_value, description, and HED.

### hed.tools.analysis.annotation_util.merge_hed_dict

**merge_hed_dict**(*sidecar_dict*, *hed_dict*)

Update a JSON sidecar based on the hed_dict values.

**Parameters**

- **sidecar_dict** (*dict*) – Dictionary representation of a BIDS JSON sidecar.
- **hed_dict** (*dict*) – Dictionary derived from a dataframe representation of HED in sidecar.

### hed.tools.analysis.annotation_util.trim_back

**trim_back**(*tag_string*)

Return a trimmed copy of tag_string.

**Parameters**
**tag_string** (*str*) – A tag string to be trimmed.

**Returns**
A copy of tag_string that has been trimmed.

**Return type**
str

#### Notes

- The trailing blanks and commas are removed from the copy.

### hed.tools.analysis.annotation_util.trim_front

**trim_front**(*tag_string*)

Return a copy of tag_string with leading blanks and commas removed.

**Parameters**
**tag_string** (*str*) – A tag string to be trimmed.

**Returns**
A copy of tag_string that has been trimmed.

**Return type**
str

#### 3.4.1.3 hed.tools.analysis.column_name_summary

Summarizes the unique column names in a dataset.

### Classes

| |
|---|
| *ColumnNameSummary*([name]) |

### hed.tools.analysis.column_name_summary.ColumnNameSummary

**class ColumnNameSummary**(*name=''*)

Bases: object

**__init__**(*name=''*)

#### Methods

| |
|---|
| *__init__*([name]) |
| get_summary([as_json]) |
| update(name, columns) |
| update_headers(column_names) |

#### 3.4.1.4 hed.tools.analysis.event_manager

Manages events of temporal extent.

### Classes

| |
|---|
| *EventManager*(input_data, hed_schema[, ...]) |

### hed.tools.analysis.event_manager.EventManager

**class EventManager**(*input_data*, *hed_schema*, *extra_defs=None*)

Bases: object

**__init__**(*input_data*, *hed_schema*, *extra_defs=None*)

Create an event manager for an events file. Manages events of temporal extent.

**Parameters**

- **input_data** (TabularInput) – Represents an events file with its sidecar.

- **hed_schema** (HedSchema) – HED schema used in this
- **extra_defs** (DefinitionDict) – Extra definitions not included in the input_data information.

**Raises**

*HedFileError* –

- if there are any unmatched offsets.

Notes: Keeps the events of temporal extend by their starting index in events file. These events are separated from the rest of the annotations.

### Methods

| | |
|---|---|
| *__init__*(input_data, hed_schema[, extra_defs]) | Create an event manager for an events file. |
| compress_strings(list_to_compress) | |
| *get_type_defs*(types) | Return a list of definition names (lower case) that correspond to one of the specified types. |
| *str_list_to_hed*(str_list) | Create a HedString object from a list of strings. |
| *unfold_context*([remove_types]) | Unfolds the event information into hed, base, and contexts either as arrays of str or of HedString. |

**get_type_defs**(*types*)

Return a list of definition names (lower case) that correspond to one of the specified types.

**Parameters**

**types** (*list*) – List of tags that are treated as types such as 'Condition-variable'

**Returns**

List of definition names (lower-case) that correspond to the specified types

**Return type**

list

**str_list_to_hed**(*str_list*)

Create a HedString object from a list of strings.

**Parameters**

**str_list** (*list*) – A list of strings to be concatenated with commas and then converted.

**Returns**

The converted list.

**Return type**

*HedString* or None

**unfold_context**(*remove_types=[]*)

Unfolds the event information into hed, base, and contexts either as arrays of str or of HedString.

**Parameters**

**remove_types** (*list*) – List of types to remove.

**Returns**

list of str or HedString representing the information without the events of temporal extent list of str or HedString representing the onsets of the events of temporal extent list of str or HedString representing the ongoing context information.

#### 3.4.1.5 hed.tools.analysis.file_dictionary

Representation of a file dictionary keyed by entity indices.

### Classes

| | |
|---|---|
| *FileDictionary*(collection_name, file_list[, ...]) | A file dictionary keyed by entity pair indices. |

### hed.tools.analysis.file_dictionary.FileDictionary

**class FileDictionary**(*collection_name*, *file_list*, *key_indices=(0, 2)*, *separator='_'*)

Bases: `object`

A file dictionary keyed by entity pair indices.

#### Notes

- The entities are identified as 0, 1, ... depending on order in the base filename.
- The entity key-value pairs are assumed separated by '_' unless a separator is provided.

**__init__**(*collection_name*, *file_list*, *key_indices=(0, 2)*, *separator='_'*)

Create a dictionary with full paths as values.

**Parameters**

- **collection_name** (*str*) – Name of the file collection for reference.
- **file_list** (*list, None*) – List containing full paths of files of interest.
- **key_indices** (*tuple, None*) – List of order of key-value pieces to assemble for the key.
- **separator** (*str*) – Character used to separate pieces of key name.

#### Notes

- This dictionary is used for cross listing BIDS style files for different studies.
-

#### Examples

If key_indices is (0, 2), the key generated for /tmp/sub-001_task-FaceCheck_run-01_events.tsv is sub_001_run-01.

### Methods

| | |
|---|---|
| *__init__*(collection_name, file_list[, ...]) | Create a dictionary with full paths as values. |
| *create_file_dict*(file_list, key_indices, ...) | Create new dict based on key indices. |
| *get_file_path*(key) | Return file path corresponding to key. |
| *iter_files*() | Iterator over the files in this dictionary. |
| *key_diffs*(other_dict) | Return symmetric key difference with other. |
| *make_file_dict*(file_list[, key_indices, ...]) | Return a dictionary of files using entity keys. |
| *make_key*(key_string[, indices, separator]) | Create a key from specified entities. |
| *output_files*([title, logger]) | Return a string with the output of the list. |

### Attributes

| | |
|---|---|
| *file_dict* | Dictionary of path values in this dictionary. |
| *file_list* | List of path values in this dictionary. |
| *key_list* | Keys in this dictionary. |
| *name* | Name of this dictionary |

**create_file_dict**(*file_list*, *key_indices*, *separator*)

Create new dict based on key indices.

**Parameters**

- **file_list** (*list*) – Paths of the files to include.
- **key_indices** (*tuple*) – A tuple of integers representing order of entities for key.
- **separator** (*str*) – The separator used between entities to form the key.

**property file_dict**

Dictionary of path values in this dictionary.

**property file_list**

List of path values in this dictionary.

**get_file_path**(*key*)

Return file path corresponding to key.

**Parameters**
**key** (*str*) – Key used to retrieve the file path.

**Returns**
File path.

**Return type**
str

**iter_files**()

Iterator over the files in this dictionary.

**Yields**
- *str* – Key into the dictionary. - file: File path.

**key_diffs**(*other_dict*)

Return symmetric key difference with other.

**Parameters**
**other_dict** (FileDictionary) –

**Returns**
The symmetric difference of the keys in this dictionary and the other one.

**Return type**
list

**property key_list**

Keys in this dictionary.

**static make_file_dict**(*file_list*, *key_indices=(0, 2)*, *separator='_'*)

Return a dictionary of files using entity keys.

**Parameters**

- **file_list** (*list*) – Paths to files to use.
- **key_indices** (*tuple*) – Positions of entities to use for key.
- **separator** (*str*) – Separator character used to construct key.

**Returns**
Key is based on key indices and value is a full path.

**Return type**
dict

**static make_key**(*key_string*, *indices=(0, 2)*, *separator='_'*)

Create a key from specified entities.

**Parameters**

- **key_string** (*str*) – The string from which to extract the key (usually a filename or path).
- **indices** (*tuple*) – Positions of entity pairs to use as key.
- **separator** (*str*) – Separator between entity pairs in the created key.

**Returns**
The created key.

**Return type**
str

**property name**

Name of this dictionary

**output_files**(*title=None*, *logger=None*)

Return a string with the output of the list.

**Parameters**

- **title** (*None, str*) – Optional title.
- **logger** (HedLogger) – Optional HED logger for recording.

**Returns**
The dictionary in string form.

**Return type**
str

#### Notes

- The logger is updated if available.

### 3.4.1.6 hed.tools.analysis.hed_tag_counts

Counts of HED tags in a file's annotations.

#### Classes

| | |
|---|---|
| *HedTagCount*(hed_tag, file_name) | |
| *HedTagCounts*(name[, total_events]) | Counts of HED tags for a tabular file. |

#### hed.tools.analysis.hed_tag_counts.HedTagCount

**class HedTagCount**(*hed_tag*, *file_name*)

Bases: object

**__init__**(*hed_tag*, *file_name*)

Counts for a particular HedTag in particular file.

**Parameters**

- **hed_tag** (HedTag) – The HedTag to keep track of.
- **file_name** (*str*) – Name of the file associated with the tag.

#### Methods

| | |
|---|---|
| *__init__*(hed_tag, file_name) | Counts for a particular HedTag in particular file. |
| get_empty() | |
| get_info([verbose]) | |
| *get_summary*() | Return a dictionary summary of the events and files for this tag. |
| *set_value*(hed_tag) | Update the tag term value counts for a HedTag. |

**get_summary**()

Return a dictionary summary of the events and files for this tag.

**Returns**
dictionary summary of events and files that contain this tag.

**Return type**
dict

**set_value**(*hed_tag*)

Update the tag term value counts for a HedTag.

**Parameters**

**hed_tag** (HedTag *or None*) – Item to use to update the value counts.

### hed.tools.analysis.hed_tag_counts.HedTagCounts

**class HedTagCounts**(*name*, *total_events=0*)

Bases: object

Counts of HED tags for a tabular file.

**Parameters**

- **name** (*str*) – An identifier for these counts (usually the filename of the tabular file)
- **total_events** (*int*) – The total number of events in the tabular file.

**__init__**(*name*, *total_events=0*)

#### Methods

| | |
|---|---|
| *__init__*(name[, total_events]) | |
| create_template(tags) | |
| get_summary() | |
| merge_tag_dicts(other_dict) | |
| *organize_tags*(tag_template) | Organize tags into categories as specified by the tag_template. |
| *update_event_counts*(hed_string_obj, file_name) | Update the tag counts based on a hed string object. |

**organize_tags**(*tag_template*)

Organize tags into categories as specified by the tag_template.

**Parameters**

**tag_template** (*dict*) – A dictionary whose keys are titles and values are lists of HED tags (str).

**Returns**

dict - keys are tags (strings) and values are list of HedTagCount for items fitting template. list - of HedTagCount objects corresponding to tags that don't fit the template.

**update_event_counts**(*hed_string_obj*, *file_name*)

Update the tag counts based on a hed string object.

**Parameters**

- **hed_string_obj** (HedString) – The HED string whose tags should be counted.
- **file_name** (*str*) – The name of the file corresponding to these counts.

#### 3.4.1.7 hed.tools.analysis.hed_tag_manager

Manager for the HED tags in a tabular file.

### Classes

| |
|---|
| *HedTagManager*(event_manager[, remove_types]) |

### hed.tools.analysis.hed_tag_manager.HedTagManager

**class HedTagManager**(*event_manager*, *remove_types=[]*)

Bases: `object`

**__init__**(*event_manager*, *remove_types=[]*)

Create a tag manager for one tabular file.

**Parameters**

- **event_manager** (*EventManager*) – an event manager for the tabular file.
- **remove_types** (*list or None*) – List of type tags (such as condition-variable) to remove.

#### Methods

| | |
|---|---|
| *__init__*(event_manager[, remove_types]) | Create a tag manager for one tabular file. |
| get_hed_obj(hed_str[, remove_types, ...]) | |
| get_hed_objs([include_context, replace_defs]) | |

#### 3.4.1.8 hed.tools.analysis.hed_type

Manages a type variable and its associated context.

### Classes

| |
|---|
| *HedType*(event_manager, name[, type_tag]) |

### hed.tools.analysis.hed_type.HedType

**class HedType**(*event_manager*, *name*, *type_tag='condition-variable'*)

Bases: `object`

**__init__**(*event_manager*, *name*, *type_tag='condition-variable'*)

Create a variable manager for one type-variable for one tabular file.

**Parameters**

- **event_manager** (EventManager) – An event manager for the tabular file.
- **name** (*str*) – Name of the tabular file as a unique identifier.
- **type_tag** (*str*) – Lowercase short form of the tag to be managed.

**Raises**

*HedFileError* –

- On errors such as unmatched onsets or missing definitions.

#### Methods

| | |
|---|---|
| *__init__*(event_manager, name[, type_tag]) | Create a variable manager for one type-variable for one tabular file. |
| get_summary() | |
| *get_type_def_names*() | Return the type defs names |
| *get_type_factors*([type_values, factor_encoding]) | Create a dataframe with the indicated type tag values as factors. |
| *get_type_list*(type_tag, item) | Find a list of the given type tag from a HedTag, HedGroup, or HedString. |
| *get_type_value_factors*(type_value) | Return the HedTypeFactors associated with type_name or None. |
| *get_type_value_level_info*(type_value) | Return type variable corresponding to type_value. |
| get_type_value_names() | |

#### Attributes

| |
|---|
| total_events |
| type_variables |

**get_type_def_names**()

Return the type defs names

**get_type_factors**(*type_values=None*, *factor_encoding='one-hot'*)

Create a dataframe with the indicated type tag values as factors.

**Parameters**

- **type_values** (*list or None*) – A list of values of type tags for which to generate factors.

- **factor_encoding** (*str*) – Type of factor encoding (one-hot or categorical).

**Returns**
Contains the specified factors associated with this type tag.

**Return type**
DataFrame

**static get_type_list**(*type_tag*, *item*)

Find a list of the given type tag from a HedTag, HedGroup, or HedString.

**Parameters**

- **type_tag** (*str*) – a tag whose direct items you wish to remove
- **item** (HedTag *or* HedGroup) – The item from which to extract condition type_variables.

**Returns**
List of the items with this type_tag

**Return type**
list

**get_type_value_factors**(*type_value*)

Return the HedTypeFactors associated with type_name or None.

**Parameters**
**type_value** (*str*) – The tag corresponding to the type's value (such as the name of the condition variable).

**Returns**
HedTypeFactors or None

**get_type_value_level_info**(*type_value*)

Return type variable corresponding to type_value.

**Parameters**
**type_value** (*str*) –

Returns:

#### 3.4.1.9 hed.tools.analysis.hed_type_counts

Manages the counts of tags such as Condition-variable and task.

### Classes

| | |
|---|---|
| *HedTypeCount*(type_value, type_tag[, file_name]) | Keeps a summary of one value of one type of variable. |
| *HedTypeCounts*(name, type_tag) | Keeps a summary of tag counts for a file. |

### hed.tools.analysis.hed_type_counts.HedTypeCount

**class HedTypeCount**(*type_value*, *type_tag*, *file_name=None*)

Bases: object

Keeps a summary of one value of one type of variable.

**Parameters**

- **type_value** (*str*) – The value of the variable to be counted
- **type_tag** (*str*) – The type of variable.

#### Examples

HedTypeCounts('SymmetricCond', 'condition-variable') keeps counts of Condition-variable/Symmetric

**__init__**(*type_value*, *type_tag*, *file_name=None*)

#### Methods

| | |
|---|---|
| *__init__*(type_value, type_tag[, file_name]) | |
| get_summary() | |
| to_dict() | |
| *update*(type_sum, file_id) | Update the counts from a HedTypeValues. |

**update**(*type_sum*, *file_id*)

Update the counts from a HedTypeValues.

**Parameters**

- **type_sum** (*dict*) – Information about the contents for a particular data file.
- **file_id** (*str or None*) – Name of the file associated with the counts.

### hed.tools.analysis.hed_type_counts.HedTypeCounts

**class HedTypeCounts**(*name*, *type_tag*)

Bases: object

Keeps a summary of tag counts for a file.

**__init__**(*name*, *type_tag*)

### Methods

| | |
|---|---|
| *__init__*(name, type_tag) | |
| *add_descriptions*(type_defs) | Update this summary based on the type variable map. |
| get_summary() | |
| update(counts) | |
| *update_summary*(type_sum[, total_events, file_id]) | Update this summary based on the type variable map. |

**add_descriptions**(*type_defs*)

Update this summary based on the type variable map.

**Parameters**

**type_defs** (HedTypeDefs) – Contains the information about the value of a type.

**update_summary**(*type_sum*, *total_events=0*, *file_id=None*)

Update this summary based on the type variable map.

**Parameters**

- **type_sum** (*dict*) – Contains the information about the value of a type.
- **total_events** (*int*) – Total number of events processed.
- **file_id** (*str*) – Unique identifier for the associated file.

#### 3.4.1.10 hed.tools.analysis.hed_type_defs

Manages definitions associated with a type such as condition-variable.

### Classes

| | |
|---|---|
| *HedTypeDefs*(definitions[, type_tag]) | Properties: |

### hed.tools.analysis.hed_type_defs.HedTypeDefs

**class HedTypeDefs**(*definitions*, *type_tag='condition-variable'*)

Bases: object

**Properties:**

**def_map (dict): keys are definition names, values are dict {type_values, description, tags}**

Example: A definition 'famous-face-cond' with contents `(Condition-variable/Face-type,Description/A face that should be recognized by the

participants,(Image,(Face,Famous)))`

would have type_values ['face_type']. All items are strings not objects.

**__init__**(*definitions*, *type_tag='condition-variable'*)

Create a definition manager for a type of variable.

**Parameters**

- **definitions** (*dict or DefinitionDict*) – A dictionary of DefinitionEntry objects.
- **type_tag** (*str*) – Lower-case HED tag string representing the type managed.

### Methods

| | |
|---|---|
| *__init__*(definitions[, type_tag]) | Create a definition manager for a type of variable. |
| *extract_def_names*(item[, no_value]) | Return a list of Def values in item. |
| *get_type_values*(item) | Return a list of type_tag values in item. |
| *split_name*(name[, lowercase]) | Split a name/# or name/x into name, x. |

### Attributes

| | |
|---|---|
| *type_def_names* | List of names of definition that have this type-variable. |
| *type_names* | List of names of the type-variables associated with type definitions. |

**static extract_def_names**(*item*, *no_value=True*)

Return a list of Def values in item.

**Parameters**

- **item** (*HedTag, HedGroup, or HedString*) – An item containing a def tag.
- **no_value** (*bool*) – If True, strip off extra values after the definition name.

**Returns**

A list of definition names (as strings).

**Return type**

list

**get_type_values**(*item*)

Return a list of type_tag values in item.

**Parameters**

**item** (*HedTag, HedGroup, or HedString*) – An item potentially containing def tags.

**Returns**

A list of the unique values associated with this type

**Return type**

list

**static split_name**(*name*, *lowercase=True*)

Split a name/# or name/x into name, x.

**Parameters**

- **name** (*str*) – The extension or value portion of a tag

- **lowercase** (*bool*) – If True

**Returns**
name of the definition str: value of the definition if it has one

**Return type**
str

**property type_def_names**

List of names of definition that have this type-variable.

**Returns**
definition names that have this type.

**Return type**
list

**property type_names**

List of names of the type-variables associated with type definitions.

**Returns**
type names associated with the type definitions

**Return type**
list

#### 3.4.1.11 hed.tools.analysis.hed_type_factors

Manages factor information for a tabular file.

### Classes

| | |
|---|---|
| *HedTypeFactors*(type_tag, type_value, ...) | Holds index of positions for a variable type for one tabular file. |

### hed.tools.analysis.hed_type_factors.HedTypeFactors

**class HedTypeFactors**(*type_tag*, *type_value*, *number_elements*)

Bases: object

Holds index of positions for a variable type for one tabular file.

**__init__**(*type_tag*, *type_value*, *number_elements*)

Constructor for HedTypeFactors.

**Parameters**

- **type_tag** (*str*) – Lowercase string corresponding to a HED tag which has a takes value child.
- **type_value** (*str*) – The value of the type summarized by this class.
- **number_elements** (*int*) – Number of elements in the data column

### Methods

| | |
|---|---|
| *__init__*(type_tag, type_value, number_elements) | Constructor for HedTypeFactors. |
| *get_factors*([factor_encoding]) | Return a DataFrame of factor vectors for this type factor. |
| get_summary() | |

### Attributes

| |
|---|
| ALLOWED_ENCODINGS |

**get_factors**(*factor_encoding='one-hot'*)

Return a DataFrame of factor vectors for this type factor.

**Parameters**
**factor_encoding** (*str*) – Specifies type of factor encoding (one-hot or categorical).

**Returns**
DataFrame containing the factor vectors as the columns.

**Return type**
DataFrame

#### 3.4.1.12 hed.tools.analysis.hed_type_manager

Manager for type factors and type definitions.

### Classes

| |
|---|
| *HedTypeManager*(event_manager) |

### hed.tools.analysis.hed_type_manager.HedTypeManager

**class HedTypeManager**(*event_manager*)

Bases: object

**__init__**(*event_manager*)

Create a variable manager for one tabular file for all type variables.

**Parameters**
**event_manager** (EventManager) – an event manager for the tabular file.

**Raises**
*HedFileError* –

- On errors such as unmatched onsets or missing definitions.

### Methods

| | |
|---|---|
| *__init__*(event_manager) | Create a variable manager for one tabular file for all type variables. |
| add_type(type_name) | |
| *get_factor_vectors*(type_tag[, type_values, ...]) | Return a DataFrame of factor vectors for the indicated HED tag and values |
| *get_type*(type_tag) | **param type_tag**<br>HED tag to retrieve the type for |
| get_type_def_names(type_var) | |
| *get_type_tag_factor*(type_tag, type_value) | Return the HedTypeFactors a specified value and extension. |
| summarize_all([as_json]) | |

### Attributes

| |
|---|
| types |

**get_factor_vectors**(*type_tag*, *type_values=None*, *factor_encoding='one-hot'*)

Return a DataFrame of factor vectors for the indicated HED tag and values

**Parameters**

- **type_tag** (*str*) – HED tag to retrieve factors for.
- **type_values** (*list or None*) – The values of the tag to create factors for or None if all unique values.
- **factor_encoding** (*str*) – Specifies type of factor encoding (one-hot or categorical).

**Returns**
DataFrame containing the factor vectors as the columns.

**Return type**
DataFrame or None

**get_type**(*type_tag*)

**Parameters**
**type_tag** (*str*) – HED tag to retrieve the type for

**Returns**
the values associated with this type tag

**Return type**
*HedType* or None

**get_type_tag_factor**(*type_tag*, *type_value*)

Return the HedTypeFactors a specified value and extension.

**Parameters**

- **type_tag** (*str or None*) – HED tag for the type
- **type_value** (*str or None*) – Value of this tag to return the factors for.

#### 3.4.1.13 hed.tools.analysis.key_map

A map of column value keys into new column values.

### Classes

| | |
|---|---|
| *KeyMap*(key_cols[, target_cols, name]) | A map of unique column values for remapping columns. |

### hed.tools.analysis.key_map.KeyMap

**class KeyMap**(*key_cols*, *target_cols=None*, *name=''*)

Bases: object

A map of unique column values for remapping columns.

**key_cols**

A list of column names that will be hashed into the keys for the map.

**Type**
list

**target_cols**

Optional list of column names that will be inserted into data and later remapped.

**Type**
list or None

**name**

An optional name of this remap for identification purposes.

**Type**
str

Notes: This mapping converts all columns in the mapping to strings. The remapping does not support other types of columns.

**__init__**(*key_cols*, *target_cols=None*, *name=''*)

Information for remapping columns of tabular files.

**Parameters**

- **key_cols** (*list*) – List of columns to be replaced (assumed in the DataFrame)
- **target_cols** (*list*) – List of replacement columns (assumed to not be in the DataFrame)
- **name** (*str*) – Name associated with this remap (usually a pathname of the events file).

### Methods

| | |
|---|---|
| *__init__*(key_cols[, target_cols, name]) | Information for remapping columns of tabular files. |
| *make_template*([additional_cols, show_counts]) | Return a dataframe template. |
| *remap*(data) | Remap the columns of a dataframe or columnar file. |
| *remove_quotes*(df[, columns]) | Remove quotes from the specified columns and convert to string. |
| *resort*() | Sort the col_map in place by the key columns. |
| *update*(data[, allow_missing]) | Update the existing map with information from data. |

### Attributes

| |
|---|
| columns |

**make_template**(*additional_cols=None*, *show_counts=True*)

Return a dataframe template.

**Parameters**

- **additional_cols** (*list or None*) – Optional list of additional columns to append to the returned dataframe.
- **show_counts** (*bool*) – If true, number of times each key combination appears is in first column

**Returns**
A dataframe containing the template.

**Return type**
DataFrame

**Raises**
*HedFileError* –

- If additional columns are not disjoint from the key columns.

### Notes

- The template consists of the unique key columns in this map plus additional columns.

**remap**(*data*)

Remap the columns of a dataframe or columnar file.

**Parameters**
**data** (*DataFrame, str*) – Columnar data (either DataFrame or filename) whose columns are to be remapped.

**Returns**

- DataFrame: New dataframe with columns remapped.
- list: List of row numbers that had no correspondence in the mapping.

**Return type**
tuple

**Raises**
*HedFileError* –

- If data is missing some of the key columns.

**static remove_quotes**(*df*, *columns=None*)

Remove quotes from the specified columns and convert to string.

**Parameters**

- **df** (*Dataframe*) – Dataframe to process by removing quotes.
- **columns** (*list*) – List of column names. If None, all columns are used.

### Notes

- Replacement is done in place.

**resort**()

Sort the col_map in place by the key columns.

**update**(*data*, *allow_missing=True*)

Update the existing map with information from data.

**Parameters**

- **data** (*DataFrame or str*) – DataFrame or filename of an events file or event map.
- **allow_missing** (*bool*) – If true allow missing keys and add as n/a columns.

**Raises**
*HedFileError* –

- If there are missing keys and allow_missing is False.

#### 3.4.1.14 hed.tools.analysis.tabular_summary

Summarize the contents of tabular files.

### Classes

| | |
|---|---|
| *TabularSummary*([value_cols, skip_cols, name]) | Summarize the contents of tabular files. |

### hed.tools.analysis.tabular_summary.TabularSummary

**class TabularSummary**(*value_cols=None*, *skip_cols=None*, *name=''*)

Bases: object

Summarize the contents of tabular files.

**__init__**(*value_cols=None*, *skip_cols=None*, *name=''*)

Constructor for a BIDS tabular file summary.

**Parameters**

- **value_cols** (*list, None*) – List of columns to be treated as value columns.

- **skip_cols** (*list, None*) – List of columns to be skipped.
- **name** (*str*) – Name associated with the dictionary.

### Methods

| | |
|---|---|
| *__init__*([value_cols, skip_cols, name]) | Constructor for a BIDS tabular file summary. |
| *extract_sidecar_template*() | Extract a BIDS sidecar-compatible dictionary. |
| *extract_summary*(summary_info) | Create a TabularSummary object from a serialized summary |
| *get_columns_info*(dataframe[, skip_cols]) | Extract unique value counts for columns. |
| *get_number_unique*([column_names]) | Return the number of unique values in columns. |
| get_summary([as_json]) | |
| *make_combined_dicts*(file_dictionary[, skip_cols]) | Return combined and individual summaries. |
| *update*(data[, name]) | Update the counts based on data. |
| *update_summary*(tab_sum) | Add TabularSummary values to this object. |

**extract_sidecar_template**()

Extract a BIDS sidecar-compatible dictionary.

**static extract_summary**(*summary_info*)

Create a TabularSummary object from a serialized summary

**Parameters**

**summary_info** (*dict or str*) – A JSON string or a dictionary containing contents of a TabularSummary.

**Returns**

contains the information in summary_info as a TabularSummary object.

**Return type**

*TabularSummary*

**static get_columns_info**(*dataframe*, *skip_cols=None*)

Extract unique value counts for columns.

**Parameters**

- **dataframe** (*DataFrame*) – The DataFrame to be analyzed.
- **skip_cols** (*list*) – List of names of columns to be skipped in the extraction.

**Returns**

**A dictionary with keys that are column names and values that**
are dictionaries of unique value counts.

**Return type**

dict

**get_number_unique**(*column_names=None*)

Return the number of unique values in columns.

**Parameters**

**column_names** (*list, None*) – A list of column names to analyze or all columns if None.

**Returns**
Column names are the keys and the number of unique values in the column are the values.

**Return type**
dict

**static make_combined_dicts**(*file_dictionary*, *skip_cols=None*)

Return combined and individual summaries.

**Parameters**

- **file_dictionary** (FileDictionary) – Dictionary of file name keys and full path.
- **skip_cols** (*list*) – Name of the column.

**Returns**

- TabularSummary: Summary of the file dictionary.
- dict: of individual TabularSummary objects.

**Return type**
tuple

**update**(*data*, *name=None*)

Update the counts based on data.

**Parameters**

- **data** (*DataFrame, str, or list*) – DataFrame containing data to update.
- **name** (*str*) – Name of the summary

**update_summary**(*tab_sum*)

Add TabularSummary values to this object.

**Parameters**
**tab_sum** (TabularSummary) – A TabularSummary to be combined.

#### Notes

- The value_cols and skip_cols are updated as long as they are not contradictory.
- A new skip column cannot be used.

#### 3.4.1.15 hed.tools.analysis.temporal_event

#### Classes

| | |
|---|---|
| *TemporalEvent*(contents, start_index, start_time) | Represents an event process with starting and ending. |

#### hed.tools.analysis.temporal_event.TemporalEvent

**class TemporalEvent**(*contents*, *start_index*, *start_time*)

Bases: object

Represents an event process with starting and ending.

Note: the contents have the Onset and duration removed.

**__init__**(*contents*, *start_index*, *start_time*)

##### Methods

| | |
|---|---|
| *__init__*(contents, start_index, start_time) | |
| set_end(end_index, end_time) | |

### 3.4.2 hed.tools.bids

Models for BIDS datasets and files.

#### Modules

| | |
|---|---|
| *hed.tools.bids.bids_dataset* | The contents of a BIDS dataset. |
| *hed.tools.bids.bids_file* | Models a BIDS file. |
| *hed.tools.bids.bids_file_dictionary* | A dictionary of BIDS files keyed to entity-value pairs. |
| *hed.tools.bids.bids_file_group* | A group of BIDS files with specified suffix name. |
| *hed.tools.bids.bids_sidecar_file* | Container for a BIDS sidecar file. |
| *hed.tools.bids.bids_tabular_dictionary* | A dictionary of tabular files keyed to BIDS entities. |
| *hed.tools.bids.bids_tabular_file* | A BIDS tabular file including its associated sidecar. |

#### 3.4.2.1 hed.tools.bids.bids_dataset

The contents of a BIDS dataset.

#### Classes

| | |
|---|---|
| *BidsDataset*(root_path[, schema, ...]) | A BIDS dataset representation primarily focused on HED evaluation. |

### hed.tools.bids.bids_dataset.BidsDataset

**class BidsDataset**(*root_path*, *schema=None*, *tabular_types=None*, *exclude_dirs=['sourcedata', 'derivatives', 'code', 'stimuli']*)

Bases: `object`

A BIDS dataset representation primarily focused on HED evaluation.

**root_path**

Real root path of the BIDS dataset.

**Type**
str

**schema**

The schema used for evaluation.

**Type**
*HedSchema* or *HedSchemaGroup*

**tabular_files**

A dictionary of BidsTabularDictionary objects containing a given type.

**Type**
dict

**__init__**(*root_path*, *schema=None*, *tabular_types=None*, *exclude_dirs=['sourcedata', 'derivatives', 'code', 'stimuli']*)

Constructor for a BIDS dataset.

**Parameters**

- **root_path** (*str*) – Root path of the BIDS dataset.
- **schema** (*HedSchema or HedSchemaGroup*) – A schema that overrides the one specified in dataset.
- **tabular_types** (*list or None*) – List of strings specifying types of tabular types to include. If None or empty, then [‘events’] is assumed.
- **exclude_dirs=['sourcedata'** –
- **'derivatives'** –
- **'code']** –

#### Methods

| | |
|---|---|
| *__init__*(root_path[, schema, tabular_types, ...]) | Constructor for a BIDS dataset. |
| *get_summary*() | Return an abbreviated summary of the dataset. |
| *get_tabular_group*([obj_type]) | Return the specified tabular file group. |
| *validate*([types, check_for_warnings]) | Validate the specified file group types. |

**get_summary**()

Return an abbreviated summary of the dataset.

**get_tabular_group**(*obj_type='events'*)

Return the specified tabular file group.

**Parameters**
**obj_type** (*str*) – Suffix of the BidsFileGroup to be returned.

**Returns**
The requested tabular group.

**Return type**
*BidsFileGroup* or None

**validate**(*types=None*, *check_for_warnings=True*)

Validate the specified file group types.

**Parameters**

- **types** (*list*) – A list of strings indicating the file group types to be validated.
- **check_for_warnings** (*bool*) – If True, check for warnings.

**Returns**
List of issues encountered during validation. Each issue is a dictionary.

**Return type**
list

#### 3.4.2.2 hed.tools.bids.bids_file

Models a BIDS file.

##### Classes

| | |
|---|---|
| *BidsFile*(file_path) | A BIDS file with entity dictionary. |

##### hed.tools.bids.bids_file.BidsFile

**class BidsFile**(*file_path*)

Bases: object

A BIDS file with entity dictionary.

**file_path**

Real path of the file.

**Type**
str

**suffix**

Suffix part of the filename.

**Type**
str

**ext**

Extension (including the .).

**Type**
str

**entity_dict**

Dictionary of entity-names (keys) and entity-values (values).

**Type**
dict

**sidecar**

Merged sidecar for this file.

**Type**
*BidsSidecarFile*

## Notes

- This class may hold the merged sidecar giving metadata for this file as well as contents.

**__init__**(*file_path*)

Constructor for a file path.

**Parameters**
**file_path** (*str*) – Full path of the file.

## Methods

| | |
|---|---|
| *__init__*(file_path) | Constructor for a file path. |
| *clear_contents*() | Set the contents attribute of this object to None. |
| get_entity(entity_name) | |
| *get_key*([entities]) | Return a key for this BIDS file given a list of entities. |
| *set_contents*([content_info, overwrite]) | Set the contents of this object. |

## Attributes

| | |
|---|---|
| *contents* | Return the current contents of this object. |

**clear_contents**()

Set the contents attribute of this object to None.

**property contents**

Return the current contents of this object.

**get_key**(*entities=None*)

Return a key for this BIDS file given a list of entities.

**Parameters**
**entities** (*tuple*) – A tuple of strings representing entities.

**Returns**
A key based on this object.

**Return type**
str

#### Notes

If entities is None, then the file path is used as the key

`set_contents`(*content_info=None*, *overwrite=False*)
Set the contents of this object.

**Parameters**

- **content_info** – The contents appropriate for this object.
- **overwrite** (*bool*) – If False and the contents are not empty, do nothing.

#### Notes

- Do not set if the contents are already set and no_overwrite is True.

### 3.4.2.3 hed.tools.bids.bids_file_dictionary

A dictionary of BIDS files keyed to entity-value pairs.

#### Classes

| | |
|---|---|
| *BidsFileDictionary*(collection_name, files[, ...]) | A dictionary of BidsFile keyed by entity pairs. |

#### hed.tools.bids.bids_file_dictionary.BidsFileDictionary

`class BidsFileDictionary`(*collection_name*, *files*, *entities=('sub', 'ses', 'task', 'run')*)
Bases: *FileDictionary*

A dictionary of BidsFile keyed by entity pairs.

The keys are simplified entity key-value pairs and the values are BidsFile objects.

`__init__`(*collection_name*, *files*, *entities=('sub', 'ses', 'task', 'run')*)
Create the dictionary keyed to entities.

**Parameters**

- **collection_name** (*str*) – Name of this collection.
- **files** (*list or dict*) – Full paths of files to include.
- **entities** (*tuple*) – Entity names to use in creating the keys.

**Raises**
*HedFileError* –

- If files has inappropriate values.

### Notes

- This function is used for cross listing BIDS style files for different studies.

### Examples

If entities is ('sub', 'ses', 'task', 'run'), a typical key might be sub-001_ses-01_task-memory_run-01.

## Methods

| | |
|---|---|
| *__init__*(collection_name, files[, entities]) | Create the dictionary keyed to entities. |
| *create_file_dict*(file_list, key_indices, ...) | Create new dict based on key indices. |
| *get_file_path*(key) | Return the file path corresponding to key. |
| *get_new_dict*(name, files) | Create a dictionary with these files. |
| *iter_files*() | Iterator over the files in this dictionary. |
| *key_diffs*(other_dict) | Return the symmetric key difference with other. |
| *make_dict*(files, entities) | Make a dictionary from files or a dict. |
| *make_file_dict*(file_list[, key_indices, ...]) | Return a dictionary of files using entity keys. |
| *make_key*(key_string[, indices, separator]) | Create a key from specified entities. |
| *make_query*([query_dict]) | Return a dictionary of files matching query. |
| *match_query*(query_dict, entity_dict) | Return True if query has a match in dictionary. |
| *output_files*([title, logger]) | Return a string with the output of the list. |
| *split_by_entity*(entity) | Split this dictionary based on an entity. |

## Attributes

| | |
|---|---|
| *file_dict* | Dictionary of keys and paths. |
| *file_list* | Paths of the files in the list. |
| *key_list* | The dictionary keys. |
| *name* | Name of this dictionary |

**create_file_dict**(*file_list*, *key_indices*, *separator*)

Create new dict based on key indices.

**Parameters**

- **file_list** (*list*) – Paths of the files to include.
- **key_indices** (*tuple*) – A tuple of integers representing order of entities for key.
- **separator** (*str*) – The separator used between entities to form the key.

**property file_dict**

Dictionary of keys and paths.

**property file_list**

Paths of the files in the list.

**get_file_path**(*key*)

Return the file path corresponding to key.

**Parameters**

**key** (*str*) – The key to use to look up the file in this dictionary.

**Returns**
The real path of the file being looked up.

**Return type**
str

### Notes

- None is returned if the key is not present.

`get_new_dict`(*name*, *files*)

Create a dictionary with these files.

**Parameters**

- **name** (*str*) – Name of this dictionary
- **files** (*list or dict*) – List or dictionary of files. These could be paths or objects.

**Returns**
The newly created dictionary.

**Return type**
*BidsFileDictionary*

### Notes

- The new dictionary uses the same type of entities for keys as this dictionary.

`iter_files`()

Iterator over the files in this dictionary.

**Yields**
*tuple* – - str: The next entity-based key. - BidsFile: The next BidsFile.

`key_diffs`(*other_dict*)

Return the symmetric key difference with other.

**Parameters**
**other_dict** (FileDictionary) –

**Returns**
The symmetric difference of the keys in this dictionary and the other one.

**Return type**
list

`property key_list`

The dictionary keys.

`make_dict`(*files*, *entities*)

Make a dictionary from files or a dict.

**Parameters**

- **files** (*list or dict*) – List or dictionary of file-like objs to use.
- **entities** (*tuple*) – Tuple of entity names to use as keys, e.g. (‘sub’, ‘run’).

**Returns**
A dictionary whose keys are entity keys and values are BidsFile objects.

**Return type**
dict

**Raises**
*HedFileError* –

- If incorrect format is passed or something not recognizable as a Bids file.

**static make_file_dict**(*file_list*, *key_indices=(0, 2)*, *separator='_'*)

Return a dictionary of files using entity keys.

**Parameters**

- **file_list** (*list*) – Paths to files to use.
- **key_indices** (*tuple*) – Positions of entities to use for key.
- **separator** (*str*) – Separator character used to construct key.

**Returns**
Key is based on key indices and value is a full path.

**Return type**
dict

**static make_key**(*key_string*, *indices=(0, 2)*, *separator='_'*)

Create a key from specified entities.

**Parameters**

- **key_string** (*str*) – The string from which to extract the key (usually a filename or path).
- **indices** (*tuple*) – Positions of entity pairs to use as key.
- **separator** (*str*) – Separator between entity pairs in the created key.

**Returns**
The created key.

**Return type**
str

**make_query**(*query_dict={'sub': '*'}*)

Return a dictionary of files matching query.

**Parameters**
**query_dict** (*dict*) – A dictionary whose keys are entities and whose values are entity values to match.

**Returns**
A dictionary entries in this dictionary that match the query.

**Return type**
dict

### Notes

- A query dictionary key a valid BIDS entity name such as sub or task.
- A query dictionary value may be a string or a list.
- A query value string should contain a specific value of the entity or a '*' indicating any value matches.
- A query value list should be a list of valid values for the corresponding entity.

**static match_query**(*query_dict*, *entity_dict*)

Return True if query has a match in dictionary.

**Parameters**

- **query_dict** (*dict*) – A dictionary representing a query about entities.
- **entity_dict** (*dict*) – A dictionary containing the entity representation for a BIDS file.

**Returns**
True if the query matches the entities representing the file.

**Return type**
bool

### Notes

- A query is a dictionary whose keys are entity names and whose values are specific entity values or '*'.

### Examples

{'sub', '001', 'run', '*'} requests all runs from subject 001.

**property name**

Name of this dictionary

**output_files**(*title=None*, *logger=None*)

Return a string with the output of the list.

**Parameters**

- **title** (*None, str*) – Optional title.
- **logger** (HedLogger) – Optional HED logger for recording.

**Returns**
The dictionary in string form.

**Return type**
str

#### Notes

- The logger is updated if available.

**split_by_entity**(*entity*)

Split this dictionary based on an entity.

**Parameters**
**entity** (*str*) – Entity name (for example task).

**Returns**

- dict: A dictionary unique values of entity as keys and BidsFileDictionary objs as values.
- dict: A BidsFileDictionary containing the files that don't have entity in their names.

**Return type**
tuple

#### Notes

- This function is used for analysis where a single subject or single type of task is being analyzed.

### 3.4.2.4 hed.tools.bids.bids_file_group

A group of BIDS files with specified suffix name.

#### Classes

| | |
|---|---|
| *BidsFileGroup*(root_path[, suffix, obj_type, ...]) | Container for BIDS files with a specified suffix. |

#### hed.tools.bids.bids_file_group.BidsFileGroup

**class BidsFileGroup**(*root_path*, *suffix='_events'*, *obj_type='tabular'*, *exclude_dirs=['sourcedata', 'derivatives', 'code', 'stimuli']*)

Bases: object

Container for BIDS files with a specified suffix.

**root_path**

Real root path of the Bids dataset.

**Type**
str

**suffix**

The file suffix specifying the class of file represented in this group (e.g., events).

**Type**
str

**obj_type**

Type of file in this group (e.g., Tabular or Timeseries).

**Type**
str

**sidecar_dict**

A dictionary of sidecars associated with this suffix .

**Type**
dict

**datafile_dict**

A dictionary with values either BidsTabularFile or BidsTimeseriesFile.

**Type**
dict

**sidecar_dir_dict**

Dictionary whose keys are directory paths and values are list of sidecars in the corresponding directory.

**Type**
dict

**__init__**(*root_path*, *suffix='_events'*, *obj_type='tabular'*, *exclude_dirs=['sourcedata', 'derivatives', 'code', 'stimuli']*)

Constructor for a BidsFileGroup.

**Parameters**

- **root_path** (*str*) – Path of the root of the BIDS dataset.
- **suffix** (*str*) – Suffix indicating the type this group represents (e.g. events, or channels, etc.).
- **obj_type** (*str*) – Indicates the type of underlying file represents the contents.
- **exclude_dirs** (*list*) – Directories to exclude.

### Methods

| | |
|---|---|
| *__init__*(root_path[, suffix, obj_type, ...]) | Constructor for a BidsFileGroup. |
| *get_sidecars_from_path*(obj) | Return applicable sidecars for the object. |
| *summarize*([value_cols, skip_cols]) | Return a BidsTabularSummary of group files. |
| *validate_datafiles*(hed_schema[, ...]) | Validate the datafiles and return an error list. |
| *validate_sidecars*(hed_schema[, ...]) | Validate merged sidecars. |

**get_sidecars_from_path**(*obj*)

Return applicable sidecars for the object.

**Parameters**
**obj** (BidsTabularFile *or* BidsSidecarFile) – The BIDS file object to get the sidecars for.

**Returns**
A list of the paths for applicable sidecars for obj starting at the root.

**Return type**
list

**summarize**(*value_cols=None*, *skip_cols=None*)

Return a BidsTabularSummary of group files.

**Parameters**

- **value_cols** (*list*) – Column names designated as value columns.
- **skip_cols** (*list*) – Column names designated as columns to skip.

**Returns**
A summary of the number of values in different columns if tabular group.

**Return type**
*TabularSummary* or None

### Notes

- The columns that are not value_cols or skip_col are summarized by counting

the number of times each unique value appears in that column.

**validate_datafiles**(*hed_schema*, *extra_def_dicts=None*, *check_for_warnings=True*, *keep_contents=False*)

Validate the datafiles and return an error list.

**Parameters**

- **hed_schema** (HedSchema) – Schema to apply to the validation.
- **extra_def_dicts** (DefinitionDict) – Extra definitions that come from outside.
- **check_for_warnings** (*bool*) – If True, include warnings in the check.
- **keep_contents** (*bool*) – If True, the underlying data files are read and their contents retained.

**Returns**
A list of validation issues found. Each issue is a dictionary.

**Return type**
list

**validate_sidecars**(*hed_schema*, *extra_def_dicts=None*, *check_for_warnings=True*)

Validate merged sidecars.

**Parameters**

- **hed_schema** (HedSchema) – HED schema for validation.
- **extra_def_dicts** (DefinitionDict) – Extra definitions
- **check_for_warnings** (*bool*) – If True, include warnings in the check.

**Returns**
A list of validation issues found. Each issue is a dictionary.

**Return type**
list

#### 3.4.2.5 hed.tools.bids.bids_sidecar_file

Container for a BIDS sidecar file.

### Classes

| | |
|---|---|
| *BidsSidecarFile*(file_path) | A BIDS sidecar file. |

### hed.tools.bids.bids_sidecar_file.BidsSidecarFile

**class BidsSidecarFile**(*file_path*)

Bases: *BidsFile*

A BIDS sidecar file.

**__init__**(*file_path*)

Constructs a bids sidecar from a file.

**Parameters**
**file_path** (*str*) – The real path of the sidecar.

#### Methods

| | |
|---|---|
| *__init__*(file_path) | Constructs a bids sidecar from a file. |
| *clear_contents*() | Set the contents attribute of this object to None. |
| get_entity(entity_name) | |
| *get_key*([entities]) | Return a key for this BIDS file given a list of entities. |
| *is_hed*(json_dict) | Return True if the json has HED. |
| *is_sidecar_for*(obj) | Return true if this is a sidecar for obj. |
| *set_contents*([content_info, overwrite]) | Set the contents of the sidecar. |

#### Attributes

| | |
|---|---|
| *contents* | Return the current contents of this object. |

**clear_contents**()

Set the contents attribute of this object to None.

**property contents**

Return the current contents of this object.

**get_key**(*entities=None*)

Return a key for this BIDS file given a list of entities.

**Parameters**
**entities** (*tuple*) – A tuple of strings representing entities.

**Returns**
A key based on this object.

**Return type**
str

### Notes

If entities is None, then the file path is used as the key

**static is_hed**(*json_dict*)

Return True if the json has HED.

**Parameters**
**json_dict** (*dict*) – A dictionary representing a JSON file or merged file.

**Returns**
True if the dictionary has HED or HED_assembled as a first or second-level key.

**Return type**
bool

**is_sidecar_for**(*obj*)

Return true if this is a sidecar for obj.

**Parameters**
**obj** (BidsFile) – A BidsFile object to check.

**Returns**
True if this is a BIDS parent of obj and False otherwise.

**Return type**
bool

### Notes

- A sidecar is a sidecar for itself.

**set_contents**(*content_info=None*, *overwrite=False*)

Set the contents of the sidecar.

**Parameters**

- **content_info** (*list, str, or None*) – If None, create a Sidecar from the object's file-path.
- **overwrite** (*bool*) – If True, overwrite contents if already set.

### Notes

- **The handling of content_info is as follows:**
  - None: This object's file_path is used.
  - str: The string is interpreted as a path of the JSON.
  - list: The list is of paths.

#### 3.4.2.6 hed.tools.bids.bids_tabular_dictionary

A dictionary of tabular files keyed to BIDS entities.

### Classes

| | |
|---|---|
| *BidsTabularDictionary*(collection_name, files) | A dictionary of tabular files keyed to BIDS entities. |

### hed.tools.bids.bids_tabular_dictionary.BidsTabularDictionary

**class BidsTabularDictionary**(*collection_name*, *files*, *entities=('sub', 'ses', 'task', 'run')*)

Bases: *BidsFileDictionary*

A dictionary of tabular files keyed to BIDS entities.

**column_dict**

Dictionary with an entity key and a list of column names for the file as the value.

**Type**
dict

**rowcount_dict**

Dictionary with an entity key and a count of number of rows for the file as the value.

**Type**
dict

**__init__**(*collection_name*, *files*, *entities=('sub', 'ses', 'task', 'run')*)

Create a dictionary of full paths.

**Parameters**

- **collection_name** (*str*) – Name of the collection.
- **files** (*list, dict*) – Contains the full paths or BidsFile representation of files of interest.
- **entities** (*tuple*) – List of indices into base file names of pieces to assemble for the key.

#### Notes

- Used for cross listing BIDS style files for different studies.

### Methods

| | |
|---|---|
| *__init__*(collection_name, files[, entities]) | Create a dictionary of full paths. |
| *count_diffs*(other_dict) | Return keys in which the number of rows differ. |
| *create_file_dict*(file_list, key_indices, ...) | Create new dict based on key indices. |
| *get_file_path*(key) | Return the file path corresponding to key. |
| *get_info*(key) | Return a dict with key, row count, and column count. |
| *get_new_dict*(name, files) | Create a new BidsTabularDictionary. |
| *iter_files*() | Iterator over the files in this dictionary. |
| *key_diffs*(other_dict) | Return the symmetric key difference with other. |
| *make_dict*(files, entities) | Make a dictionary from files or a dict. |
| *make_file_dict*(file_list[, key_indices, ...]) | Return a dictionary of files using entity keys. |
| *make_key*(key_string[, indices, separator]) | Create a key from specified entities. |
| *make_new*(name, files) | Create a dictionary with these files. |
| *make_query*([query_dict]) | Return a dictionary of files matching query. |
| *match_query*(query_dict, entity_dict) | Return True if query has a match in dictionary. |
| *output_files*([title, logger]) | Return a string with the output of the list. |
| *report_diffs*(tsv_dict[, logger]) | Reports and logs the contents and differences between this tabular dictionary and another |
| set_tsv_info() | |
| *split_by_entity*(entity) | Split this dictionary based on an entity. |

### Attributes

| | |
|---|---|
| *file_dict* | Dictionary of keys and paths. |
| *file_list* | Paths of the files in the list. |
| *key_list* | The dictionary keys. |
| *name* | Name of this dictionary |

**count_diffs**(*other_dict*)

Return keys in which the number of rows differ.

**Parameters**
**other_dict** (FileDictionary) – A file dictionary object.

**Returns**
A list containing 3-element tuples.

**Return type**
list

### Notes

- **The returned tuples consist of**
  - str: The key representing the file.
  - int: Number of rows in the file in this dictionary.
  - int: Number of rows in the file in the other dictionary.

`create_file_dict`(*file_list*, *key_indices*, *separator*)

Create new dict based on key indices.

**Parameters**

- **file_list** (*list*) – Paths of the files to include.
- **key_indices** (*tuple*) – A tuple of integers representing order of entities for key.
- **separator** (*str*) – The separator used between entities to form the key.

`property file_dict`

Dictionary of keys and paths.

`property file_list`

Paths of the files in the list.

`get_file_path`(*key*)

Return the file path corresponding to key.

**Parameters**
**key** (*str*) – The key to use to look up the file in this dictionary.

**Returns**
The real path of the file being looked up.

**Return type**
str

### Notes

- None is returned if the key is not present.

`get_info`(*key*)

Return a dict with key, row count, and column count.

**Parameters**
**key** (*str*) – The key for file whose information is to be returned.

**Returns**
A dictionary with key, row_count, and columns entries.

**Return type**
dict

`get_new_dict`(*name*, *files*)

Create a new BidsTabularDictionary.

**Parameters**

- **name** (*str*) – Name of the new object.

- **files** (*list, dict*) – List or dictionary specifying the files to include.

**Returns**
The object contains just the specified files.

**Return type**
*BidsTabularDictionary*

### Notes

- The created object uses the entities from this object

**iter_files**()

Iterator over the files in this dictionary.

**Yields**
*tuple* – - str: The next key. - BidsTabularFile: The next object. - int: Number of rows - list: List of column names

**key_diffs**(*other_dict*)

Return the symmetric key difference with other.

**Parameters**
**other_dict** (FileDictionary) –

**Returns**
The symmetric difference of the keys in this dictionary and the other one.

**Return type**
list

**property key_list**

The dictionary keys.

**make_dict**(*files*, *entities*)

Make a dictionary from files or a dict.

**Parameters**

- **files** (*list or dict*) – List or dictionary of file-like objs to use.
- **entities** (*tuple*) – Tuple of entity names to use as keys, e.g. ('sub', 'run').

**Returns**
A dictionary whose keys are entity keys and values are BidsFile objects.

**Return type**
dict

**Raises**
*HedFileError* –

- If incorrect format is passed or something not recognizable as a Bids file.

**static make_file_dict**(*file_list*, *key_indices=(0, 2)*, *separator='_'*)

Return a dictionary of files using entity keys.

**Parameters**

- **file_list** (*list*) – Paths to files to use.
- **key_indices** (*tuple*) – Positions of entities to use for key.

- **separator** (*str*) – Separator character used to construct key.

**Returns**
Key is based on key indices and value is a full path.

**Return type**
dict

**static make_key**(*key_string*, *indices=(0, 2)*, *separator='_'*)

Create a key from specified entities.

**Parameters**

- **key_string** (*str*) – The string from which to extract the key (usually a filename or path).
- **indices** (*tuple*) – Positions of entity pairs to use as key.
- **separator** (*str*) – Separator between entity pairs in the created key.

**Returns**
The created key.

**Return type**
str

**make_new**(*name*, *files*)

Create a dictionary with these files.

**Parameters**

- **name** (*str*) – Name of this dictionary
- **files** (*list or dict*) – List or dictionary of files. These could be paths or objects.

**Returns**
The newly created dictionary.

**Return type**
*BidsTabularDictionary*

**make_query**(*query_dict={'sub': '*'}*)

Return a dictionary of files matching query.

**Parameters**
**query_dict** (*dict*) – A dictionary whose keys are entities and whose values are entity values to match.

**Returns**
A dictionary entries in this dictionary that match the query.

**Return type**
dict

### Notes

- A query dictionary key a valid BIDS entity name such as sub or task.
- A query dictionary value may be a string or a list.
- A query value string should contain a specific value of the entity or a '*' indicating any value matches.
- A query value list should be a list of valid values for the corresponding entity.

**static match_query**(*query_dict*, *entity_dict*)

Return True if query has a match in dictionary.

**Parameters**

- **query_dict** (*dict*) – A dictionary representing a query about entities.
- **entity_dict** (*dict*) – A dictionary containing the entity representation for a BIDS file.

**Returns**
True if the query matches the entities representing the file.

**Return type**
bool

### Notes

- A query is a dictionary whose keys are entity names and whose values are specific entity values or '*'.

### Examples

{'sub', '001', 'run', '*'} requests all runs from subject 001.

**property name**

Name of this dictionary

**output_files**(*title=None*, *logger=None*)

Return a string with the output of the list.

**Parameters**

- **title** (*None, str*) – Optional title.
- **logger** (HedLogger) – Optional HED logger for recording.

**Returns**
The dictionary in string form.

**Return type**
str

#### Notes

- The logger is updated if available.

**report_diffs**(*tsv_dict*, *logger=None*)

Reports and logs the contents and differences between this tabular dictionary and another

**Parameters**

- **tsv_dict** (BidsTabularDictionary) – A dictionary representing BIDS-keyed tsv files.
- **logger** (HedLogger) – A HedLogger object for reporting the values by key.

**Returns**
A string with the differences.

**Return type**
str

**split_by_entity**(*entity*)

Split this dictionary based on an entity.

**Parameters**
**entity** (*str*) – Entity name (for example task).

**Returns**

- dict: A dictionary unique values of entity as keys and BidsFileDictionary objs as values.
- dict: A BidsFileDictionary containing the files that don't have entity in their names.

**Return type**
tuple

#### Notes

- This function is used for analysis where a single subject or single type of task is being analyzed.

### 3.4.2.7 hed.tools.bids.bids_tabular_file

A BIDS tabular file including its associated sidecar.

#### Classes

| | |
|---|---|
| *BidsTabularFile*(file_path) | A BIDS tabular file including its associated sidecar. |

### hed.tools.bids.bids_tabular_file.BidsTabularFile

**class BidsTabularFile**(*file_path*)

Bases: *BidsFile*

A BIDS tabular file including its associated sidecar.

**__init__**(*file_path*)

Constructor for a BIDS tabular file.

**Parameters**
**file_path** (*str*) – Path of the tabular file.

#### Methods

| | |
|---|---|
| *__init__*(file_path) | Constructor for a BIDS tabular file. |
| *clear_contents*() | Set the contents attribute of this object to None. |
| get_entity(entity_name) | |
| *get_key*([entities]) | Return a key for this BIDS file given a list of entities. |
| *set_contents*([content_info, overwrite]) | Set the contents of this tabular file. |

#### Attributes

| | |
|---|---|
| *contents* | Return the current contents of this object. |

**clear_contents**()

Set the contents attribute of this object to None.

**property contents**

Return the current contents of this object.

**get_key**(*entities=None*)

Return a key for this BIDS file given a list of entities.

**Parameters**
**entities** (*tuple*) – A tuple of strings representing entities.

**Returns**
A key based on this object.

**Return type**
str

#### Notes

If entities is None, then the file path is used as the key

**set_contents**(*content_info=None*, *overwrite=False*)

Set the contents of this tabular file.

**Parameters**

- **content_info** (*None*) – This always uses the internal file_path to create the contents.
- **overwrite** – If False, do not overwrite existing contents if any.

### 3.4.3 hed.tools.remodeling

Remodeling tools for revising and summarizing tabular files.

#### Modules

| | |
|---|---|
| *hed.tools.remodeling.backup_manager* | Class to manage backups for remodeling tools. |
| *hed.tools.remodeling.cli* | Command-line interface for remodeling tools. |
| *hed.tools.remodeling.dispatcher* | Controller for applying operations to tabular files and saving the results. |
| *hed.tools.remodeling.operations* | Remodeling operations. |

#### 3.4.3.1 hed.tools.remodeling.backup_manager

Class to manage backups for remodeling tools.

#### Classes

| |
|---|
| *BackupManager*(data_root[, backups_root]) |

#### hed.tools.remodeling.backup_manager.BackupManager

**class BackupManager**(*data_root*, *backups_root=None*)

Bases: object

**__init__**(*data_root*, *backups_root=None*)

Constructor for the backup manager.

**Parameters**

- **data_root** (*str*) – Full path of the root of the data directory.
- **backups_root** (*str or None*) – Full path to the root where backups subdirectory is located.

**Raises**

*HedFileError* –

- If the data_root does not correspond to a real directory.

### Methods

| | |
|---|---|
| *__init__*(data_root[, backups_root]) | Constructor for the backup manager. |
| *create_backup*(file_list[, backup_name, verbose]) | Create a new backup from file_list. |
| *get_backup*(backup_name) | Return the dictionary corresponding to backup_name. |
| *get_backup_files*(backup_name[, original_paths]) | Returns a list of full paths of files contained in the backup. |
| *get_backup_path*(backup_name, file_name) | Retrieve the file from the backup or throw an error. |
| get_file_key(file_name) | |
| *get_task*(task_names, file_path) | Return the task if the file name contains a task_xxx where xxx is in task_names. |
| *restore_backup*([backup_name, task_names, ...]) | Restore the files from backup_name to the main directory. |

### Attributes

| |
|---|
| BACKUP_DICTIONARY |
| BACKUP_ROOT |
| DEFAULT_BACKUP_NAME |
| RELATIVE_BACKUP_LOCATION |

**create_backup**(*file_list*, *backup_name=None*, *verbose=False*)

Create a new backup from file_list.

**Parameters**

- **file_list** (*list*) – Full paths of the files to be in the backup.
- **backup_name** (*str or None*) – Name of the backup. If None, uses the default
- **verbose** (*bool*) – If True, print out the files that are being backed up.

**Returns**

True if the backup was successful. False if a backup of that name already exists.

**Return type**

bool

**Raises**

- ***HedFileError*** –
  - For missing or incorrect files.
- **OS-related error** –
  - OS-related error when file copying occurs.

**get_backup**(*backup_name*)

Return the dictionary corresponding to backup_name.

**Parameters**
**backup_name** (*str*) – Name of the backup to be retrieved.

**Returns**
The dictionary with the backup info.

### Notes

The dictionary with backup information has keys that are the paths of the backed up files relative to the backup root. The values in this dictionary are the dates on which the particular file was backed up.

**get_backup_files**(*backup_name*, *original_paths=False*)

Returns a list of full paths of files contained in the backup.

**Parameters**

- **backup_name** (*str*) – Name of the backup.
- **original_paths** (*bool*) – If true return the original paths.

**Returns**
Full paths of the original files backed (original_paths=True) or the paths in the backup.

**Return type**
list

**Raises**
*HedFileError* –

- If not backup named backup_name exists.

**get_backup_path**(*backup_name*, *file_name*)

Retrieve the file from the backup or throw an error.

**Parameters**

- **backup_name** (*str*) – Name of the backup.
- **file_name** (*str*) – Full path of the file to be retrieved.

**Returns**
Full path of the corresponding file in the backup.

**Return type**
str

**static get_task**(*task_names*, *file_path*)

Return the task if the file name contains a task_xxx where xxx is in task_names.

**Parameters**

- **task_names** (*list*) – List of task names (without the **task_** prefix).
- **file_path** (*str*) – Path of the filename to be tested.

**Returns**
the task name or '' if there is no task_xxx or xxx is not in task_names.

**Return type**
str

**restore_backup**(*backup_name='default_back'*, *task_names=[]*, *verbose=True*)

Restore the files from backup_name to the main directory.

**Parameters**

- **backup_name** (*str*) – Name of the backup to restore.
- **task_names** (*list*) – A list of task names to restore.
- **verbose** (*bool*) – If true, print out the file names being restored.

#### 3.4.3.2 hed.tools.remodeling.cli

Command-line interface for remodeling tools.

### Modules

| | |
|---|---|
| *hed.tools.remodeling.cli.run_remodel* | Main command-line program for running the remodeling tools. |
| *hed.tools.remodeling.cli.run_remodel_backup* | Command-line program for creating a backup. |
| *hed.tools.remodeling.cli.run_remodel_restore* | Command-line program for restoring files from backup. |

### hed.tools.remodeling.cli.run_remodel

Main command-line program for running the remodeling tools.

### Functions

| | |
|---|---|
| *get_parser*() | Create a parser for the run_remodel command-line arguments. |
| *main*([arg_list]) | The command-line program. |
| *parse_arguments*([arg_list]) | Parse the command line arguments or arg_list if given. |
| *run_bids_ops*(dispatch, args) | Run the remodeler on a BIDS dataset. |
| *run_direct_ops*(dispatch, args) | Run the remodeler on files of a specified form in a directory tree. |

### hed.tools.remodeling.cli.run_remodel.get_parser

**get_parser**()

Create a parser for the run_remodel command-line arguments.

**Returns**
A parser for parsing the command line arguments.

**Return type**
argparse.ArgumentParser

### hed.tools.remodeling.cli.run_remodel.main

**main**(*arg_list=None*)

The command-line program.

**Parameters**
**arg_list** (*list or None*) – Called with value None when called from the command line. Otherwise, called with the command-line parameters as an argument list.

**Raises**
*HedFileError* –

- if the data root directory does not exist.
- if the specified backup does not exist.

### hed.tools.remodeling.cli.run_remodel.parse_arguments

**parse_arguments**(*arg_list=None*)

Parse the command line arguments or arg_list if given.

**Parameters**
**arg_list** (*list*) – List of command line arguments as a list.

**Returns**
Argument object List: A list of parsed operations (each operation is a dictionary).

**Return type**
Object

**Raises**
**ValueError** –

- If the operations were unable to be correctly parsed.

### hed.tools.remodeling.cli.run_remodel.run_bids_ops

**run_bids_ops**(*dispatch*, *args*)

Run the remodeler on a BIDS dataset.

**Parameters**

- **dispatch** (Dispatcher) – Manages the execution of the operations.
- **args** (*Object*) – The command-line arguments as an object.

### hed.tools.remodeling.cli.run_remodel.run_direct_ops

**run_direct_ops**(*dispatch*, *args*)

Run the remodeler on files of a specified form in a directory tree.

**Parameters**

- **dispatch** (Dispatcher) – Controls the application of the operations and backup.
- **args** (*argparse.Namespace*) – Dictionary of arguments and their values.

### hed.tools.remodeling.cli.run_remodel_backup

Command-line program for creating a backup.

#### Functions

| | |
|---|---|
| *get_parser*() | Create a parser for the run_remodel_backup command-line arguments. |
| *main*([arg_list]) | The command-line program for making a remodel backup. |

### hed.tools.remodeling.cli.run_remodel_backup.get_parser

**get_parser**()

Create a parser for the run_remodel_backup command-line arguments.

**Returns**
A parser for parsing the command line arguments.

**Return type**
argparse.ArgumentParser

### hed.tools.remodeling.cli.run_remodel_backup.main

**main**(*arg_list=None*)

The command-line program for making a remodel backup.

**Parameters**
**arg_list** (*list or None*) – Called with value None when called from the command line. Otherwise, called with the command-line parameters as an argument list.

**Raises**
*HedFileError* –

- If the specified backup already exists.

### hed.tools.remodeling.cli.run_remodel_restore

Command-line program for restoring files from backup.

#### Functions

| | |
|---|---|
| *get_parser*() | Create a parser for the run_remodel_restore command-line arguments. |
| *main*([arg_list]) | The command-line program for restoring a remodel backup. |

### hed.tools.remodeling.cli.run_remodel_restore.get_parser

**get_parser**()

Create a parser for the run_remodel_restore command-line arguments.

**Returns**
A parser for parsing the command line arguments.

**Return type**
argparse.ArgumentParser

### hed.tools.remodeling.cli.run_remodel_restore.main

**main**(*arg_list=None*)

The command-line program for restoring a remodel backup.

**Parameters**
**arg_list** (*list or None*) – Called with value None when called from the command line. Otherwise, called with the command-line parameters as an argument list.

**Raises**
*HedFileError* –

- if the specified backup does not exist.

#### 3.4.3.3 hed.tools.remodeling.dispatcher

Controller for applying operations to tabular files and saving the results.

### Classes

| | |
|---|---|
| *Dispatcher*(operation_list[, data_root, ...]) | Controller for applying operations to tabular files and saving the results. |

### hed.tools.remodeling.dispatcher.Dispatcher

**class Dispatcher**(*operation_list*, *data_root=None*, *backup_name='default_back'*, *hed_versions=None*)

Bases: object

Controller for applying operations to tabular files and saving the results.

**__init__**(*operation_list*, *data_root=None*, *backup_name='default_back'*, *hed_versions=None*)

Constructor for the dispatcher.

**Parameters**

- **operation_list** (*list*) – List of unparsed operations.
- **data_root** (*str or None*) – Root directory for the dataset. If none, then backups are not made.
- **hed_versions** (*str, list,* HedSchema*, or* HedSchemaGroup) – The HED schema.

**Raises**

- *HedFileError* –
  - If the specified backup does not exist.
- **ValueError** –
  - If any of the operations cannot be parsed correctly.

### Methods

| | |
|---|---|
| *__init__*(operation_list[, data_root, ...]) | Constructor for the dispatcher. |
| errors_to_str(messages[, title, sep]) | |
| *get_data_file*(file_designator) | Get the correct data file give the file designator. |
| get_schema(hed_versions) | |
| *get_summaries*([file_formats]) | Return the summaries in a dictionary of strings suitable for saving or archiving. |
| *get_summary_save_dir*() | Return the directory in which to save the summaries. |
| parse_operations(operation_list) | |
| *post_proc_data*(df) | Replace all nan entries with 'n/a' for BIDS compliance |
| *prep_data*(df) | Make a copy and replace all n/a entries in the data frame by np.NaN for processing. |
| *run_operations*(file_path[, sidecar, verbose]) | Run the dispatcher operations on a file. |
| *save_summaries*([save_formats, ...]) | Save the summary files in the specified formats. |

### Attributes

| |
|---|
| REMODELING_SUMMARY_PATH |

**get_data_file**(*file_designator*)

Get the correct data file give the file designator.

**Parameters**
**file_designator** (*str, DataFrame*) – A dataFrame or the full path of the dataframe in the original dataset.

**Returns**
DataFrame after reading the path.

**Return type**
DataFrame

**Raises**
*HedFileError* –
- If a valid file cannot be found.

### Notes

- If a string is passed and there is a backup manager, the string must correspond to the full path of the file in the original dataset. In this case, the corresponding backup file is read and returned.
- If a string is passed and there is no backup manager, the data file corresponding to the file_designator is read and returned.
- If a Pandas DataFrame is passed, return a copy.

**get_summaries**(*file_formats=['.txt', '.json']*)

Return the summaries in a dictionary of strings suitable for saving or archiving.

**Parameters**
**file_formats** (*list*) – List of formats for the context files ('.json' and '.txt' are allowed).

**Returns**
A list of dictionaries of summaries keyed to filenames.

**Return type**
list

**get_summary_save_dir**()

Return the directory in which to save the summaries.

**Returns**
the data_root + remodeling summary path

**Return type**
str

**Raises**
***HedFileError*** –

- If this dispatcher does not have a data_root.

**static post_proc_data**(*df*)

Replace all nan entries with 'n/a' for BIDS compliance

**Parameters**
**df** (*DataFrame*) – The DataFrame to be processed.

**Returns**
DataFrame with the 'np.NAN replaced by 'n/a'

**Return type**
DataFrame

**static prep_data**(*df*)

Make a copy and replace all n/a entries in the data frame by np.NaN for processing.

**Parameters**
**df** (*DataFrame*) –

**run_operations**(*file_path*, *sidecar=None*, *verbose=False*)

Run the dispatcher operations on a file.

**Parameters**

- **file_path** (*str or DataFrame*) – Full path of the file to be remodeled or a DataFrame
- **sidecar** (Sidecar *or file-like*) – Only needed for HED operations.

- **verbose** (*bool*) – If true, print out progress reports

**Returns**
The processed dataframe.

**Return type**
DataFrame

**save_summaries**(*save_formats=['.json', '.txt']*, *individual_summaries='separate'*, *summary_dir=None*)

Save the summary files in the specified formats.

**Parameters**

- **save_formats** (*list*) – A list of formats [".txt", ."json"]
- **individual_summaries** (*str*) – If True, include summaries of individual files.
- **summary_dir** (*str or None*) – Directory for saving summaries.

#### Notes

The summaries are saved in the dataset derivatives/remodeling folder if no save_dir is provided.

#### 3.4.3.4 hed.tools.remodeling.operations

Remodeling operations.

### Modules

| | |
|---|---|
| *hed.tools.remodeling.operations.base_op* | Base class for remodeling operations. |
| *hed.tools.remodeling.operations.base_summary* | Abstract base class for the contents of summary operations. |
| *hed.tools.remodeling.operations.convert_columns_op* | Convert the type of the specified columns of a tabular file. |
| *hed.tools.remodeling.operations.factor_column_op* | Create tabular file factor columns from column values. |
| *hed.tools.remodeling.operations.factor_hed_tags_op* | Create tabular file factors from tag queries. |
| *hed.tools.remodeling.operations.factor_hed_type_op* | Create tabular file factors from type variables. |
| *hed.tools.remodeling.operations.merge_consecutive_op* | Merge consecutive rows with same column value. |
| *hed.tools.remodeling.operations.number_groups_op* | Implementation in progress. |
| *hed.tools.remodeling.operations.number_rows_op* | Implementation in progress. |
| *hed.tools.remodeling.operations.remap_columns_op* | Map values in m columns into a new combinations in n columns. |
| *hed.tools.remodeling.operations.remove_columns_op* | Remove columns from a tabular file. |
| *hed.tools.remodeling.operations.remove_rows_op* | Remove rows from a tabular file. |
| *hed.tools.remodeling.operations.rename_columns_op* | Rename columns in a tabular file. |
| *hed.tools.remodeling.operations.reorder_columns_op* | Reorder columns in a tabular file. |
| *hed.tools.remodeling.operations.split_rows_op* | Split rows in a tabular file into multiple rows based on a column. |
| *hed.tools.remodeling.operations.summarize_column_names_op* | Summarize the column names in a collection of tabular files. |
| *hed.tools.remodeling.operations.summarize_column_values_op* | Summarize the values in the columns of a tabular file. |
| *hed.tools.remodeling.operations.summarize_definitions_op* | Summarize the type_defs in the dataset. |
| *hed.tools.remodeling.operations.summarize_hed_tags_op* | Summarize the HED tags in collection of tabular files. |
| *hed.tools.remodeling.operations.summarize_hed_type_op* | Summarize a HED type tag in a collection of tabular files. |
| *hed.tools.remodeling.operations.summarize_hed_validation_op* | Validate the HED tags in a dataset and report errors. |
| *hed.tools.remodeling.operations.summarize_sidecar_from_events_op* | Create a JSON sidecar from column values in a collection of tabular files. |
| *hed.tools.remodeling.operations.valid_operations* | The valid operations for the remodeling tools. |

### hed.tools.remodeling.operations.base_op

Base class for remodeling operations.

### Classes

| | |
|---|---|
| *BaseOp*(op_spec, parameters) | Base class for operations. |

### hed.tools.remodeling.operations.base_op.BaseOp

**class BaseOp**(*op_spec*, *parameters*)

Bases: `object`

Base class for operations. All remodeling operations should extend this class.

The base class holds the parameters and does basic parameter checking against the operation's specification.

**__init__**(*op_spec*, *parameters*)

Base class constructor for operations.

**Parameters**

- **op_spec** (*dict*) – Specification for required and optional parameters.
- **parameters** (*dict*) – Actual values of the parameters for the operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.
- **ValueError** –
  - If the specification is missing a valid operation.

#### Methods

| | |
|---|---|
| *__init__*(op_spec, parameters) | Base class constructor for operations. |
| *check_parameters*(parameters) | Verify that the parameters meet the operation specification. |
| *do_op*(dispatcher, df, name[, sidecar]) | Base class method to be overridden by each operation. |

**check_parameters**(*parameters*)

Verify that the parameters meet the operation specification.

**Parameters**

**parameters** (*dict*) – Dictionary of parameters for this operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

**do_op**(*dispatcher*, *df*, *name*, *sidecar=None*)

Base class method to be overridden by each operation.

**Parameters**

- **dispatcher** (Dispatcher) – Manages the operation I/O.
- **df** (*DataFrame*) – The tabular file to be remodeled.
- **name** (*str*) – Unique identifier for the data – often the original file path.
- **sidecar** (Sidecar *or file-like*) – A JSON sidecar needed for HED operations.

### hed.tools.remodeling.operations.base_summary

Abstract base class for the contents of summary operations.

### Classes

| | |
|---|---|
| *BaseSummary*(sum_op) | Abstract base class for summary contents. |

### hed.tools.remodeling.operations.base_summary.BaseSummary

**class BaseSummary**(*sum_op*)

Bases: ABC

Abstract base class for summary contents. Should not be instantiated.

**Parameters**

**sum_op** (BaseOp) – Operation corresponding to this summary.

**__init__**(*sum_op*)

### Methods

| | |
|---|---|
| *__init__*(sum_op) | |
| dump_summary(filename, summary) | |
| *get_details_dict*(summary_info) | Return the summary-specific information. |
| get_individual(summary_details[, separately]) | |
| *get_summary*([individual_summaries]) | Return a summary dictionary with the information. |
| *get_summary_details*([include_individual]) | Return a dictionary with the details for individual files and the overall dataset. |
| get_text_summary([individual_summaries]) | |
| get_text_summary_details([include_individual]) | |
| *merge_all_info*() | Return merged information. |
| save(save_dir[, file_formats, ...]) | |
| *update_summary*(summary_dict) | Method to update summary for a given tabular input. |

### Attributes

| |
|---|
| DISPLAY_INDENT |
| INDIVIDUAL_SUMMARIES_PATH |

**abstract get_details_dict**(*summary_info*)

Return the summary-specific information.

**Parameters**
**summary_info** (*object*) – Summary to return info from

**Returns**
dictionary with the results.

**Return type**
dict

#### Notes

Abstract method be implemented by each individual summary.

### Notes

The expected return value is a dictionary of the form:

{"Name": "", "Total events": 0, "Total files": 0, "Files": [], "Specifics": {}}"

**get_summary**(*individual_summaries='separate'*)

Return a summary dictionary with the information.

**Parameters**
**individual_summaries** (*str*) – "separate", "consolidated", or "none"

**Returns**
dict - dictionary with "Dataset" and "Individual files" keys.

**Notes: The individual_summaries value is processed as follows**

- "separate" individual summaries are to be in separate files
- "consolidated" means that the individual summaries are in same file as overall summary
- "none" means that only the overall summary is produced.

**get_summary_details**(*include_individual=True*)

Return a dictionary with the details for individual files and the overall dataset.

**Parameters**
**include_individual** (*bool*) – If True, summaries for individual files are included.

**Returns**
dict - a dictionary with 'Dataset' and 'Individual files' keys.

### Notes

- The 'Dataset' value is either a string or a dictionary with the overall summary.
- **The 'Individual files' value is dictionary whose keys are file names and values are** their corresponding summaries.

Users are expected to provide merge_all_info and get_details_dict to support this.

**abstract merge_all_info**()

Return merged information.

**Returns**
Consolidated summary of information.

**Return type**
object

### Notes

Abstract method be implemented by each individual summary.

**abstract update_summary**(*summary_dict*)

Method to update summary for a given tabular input.

**Parameters**
**summary_dict** (*dict*) –

## hed.tools.remodeling.operations.convert_columns_op

Convert the type of the specified columns of a tabular file.

### Classes

| | |
|---|---|
| *ConvertColumnsOp*(parameters) | Convert. |

## hed.tools.remodeling.operations.convert_columns_op.ConvertColumnsOp

**class ConvertColumnsOp**(*parameters*)

Bases: *BaseOp*

Convert.

**Required remodeling parameters:**

- **column_names** (*list*): The list of columns to convert.
- **convert_to_** (*str*): Name of type to convert to. (One of 'str', 'int', 'float', 'fixed'.)
- **decimal_places** (*int*): Number decimal places to keep (for fixed only).

**__init__**(*parameters*)

Constructor for the convert columns operation.

**Parameters**
**parameters** (*dict*) – Parameter values for required and optional parameters.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.
- **ValueError** –
  - If convert_to is not one of the allowed values.

### Methods

| | |
|---|---|
| *__init__*(parameters) | Constructor for the convert columns operation. |
| *check_parameters*(parameters) | Verify that the parameters meet the operation specification. |
| *do_op*(dispatcher, df, name[, sidecar]) | Convert the specified column to a specified type. |

### Attributes

| |
|---|
| PARAMS |

**check_parameters**(*parameters*)

Verify that the parameters meet the operation specification.

**Parameters**
**parameters** (*dict*) – Dictionary of parameters for this operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

**do_op**(*dispatcher*, *df*, *name*, *sidecar=None*)

Convert the specified column to a specified type.

**Parameters**

- **dispatcher** (Dispatcher) – Manages the operation I/O.
- **df** (*DataFrame*) – The DataFrame to be remodeled.
- **name** (*str*) – Unique identifier for the dataframe – often the original file path.
- **sidecar** (Sidecar *or file-like*) – Only needed for HED operations.

**Returns**
A new DataFrame with the factor columns appended.

**Return type**
DataFrame

### hed.tools.remodeling.operations.factor_column_op

Create tabular file factor columns from column values.

### Classes

| | |
|---|---|
| *FactorColumnOp*(parameters) | Create tabular file factor columns from column values. |

### hed.tools.remodeling.operations.factor_column_op.FactorColumnOp

**class FactorColumnOp**(*parameters*)

Bases: *BaseOp*

Create tabular file factor columns from column values.

**Required remodeling parameters:**

- **column_name** (*str*): The name of a column in the DataFrame.
- **factor_values** (*list*): Values in the column column_name to create factors for.
- **factor_names** (*list*): Names to use as the factor columns.

**__init__**(*parameters*)

Constructor for the factor column operation.

**Parameters**
**parameters** (*dict*) – Parameter values for required and optional parameters.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.
- **ValueError** –
  - If factor_names is not empty and is not the same length as factor_values.

#### Methods

| | |
|---|---|
| *__init__*(parameters) | Constructor for the factor column operation. |
| *check_parameters*(parameters) | Verify that the parameters meet the operation specification. |
| *do_op*(dispatcher, df, name[, sidecar]) | Create factor columns based on values in a specified column. |

### Attributes

| PARAMS |
|---|

**check_parameters**(*parameters*)

Verify that the parameters meet the operation specification.

**Parameters**
**parameters** (*dict*) – Dictionary of parameters for this operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

**do_op**(*dispatcher*, *df*, *name*, *sidecar=None*)

Create factor columns based on values in a specified column.

**Parameters**

- **dispatcher** (Dispatcher) – Manages the operation I/O.
- **df** (*DataFrame*) – The DataFrame to be remodeled.
- **name** (*str*) – Unique identifier for the dataframe – often the original file path.
- **sidecar** (Sidecar *or file-like*) – Not needed for this operation.

**Returns**
A new DataFrame with the factor columns appended.

**Return type**
DataFrame

## hed.tools.remodeling.operations.factor_hed_tags_op

Create tabular file factors from tag queries.

### Classes

| | |
|---|---|
| *FactorHedTagsOp*(parameters) | Create tabular file factors from tag queries. |

### hed.tools.remodeling.operations.factor_hed_tags_op.FactorHedTagsOp

**class FactorHedTagsOp**(*parameters*)

Bases: *BaseOp*

Create tabular file factors from tag queries.

**Required remodeling parameters:**

- **queries** (*list*): Queries to be applied successively as filters.
- **query_names** (*list*): Column names for the query factors.
- **remove_types** (*list*): Structural HED tags to be removed.
- **expand_context** (*bool*): Expand the context if True.

#### Notes

- If factor column names are not provided, *query1*, *query2*, ... are used.
- When the context is expanded, the effect of events for temporal extent is accounted for.
- Context expansion is not implemented in the current version.

**__init__**(*parameters*)

Constructor for the factor HED tags operation.

**Parameters**
**parameters** (*dict*) – Actual values of the parameters for the operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.
- **ValueError** –
  - If the specification is missing a valid operation.
  - If the length of query names is not empty and not same length as queries.
  - If there are duplicate query names.

#### Methods

| | |
|---|---|
| *__init__*(parameters) | Constructor for the factor HED tags operation. |
| *check_parameters*(parameters) | Verify that the parameters meet the operation specification. |
| *do_op*(dispatcher, df, name[, sidecar]) | Factor the column using HED tag queries. |

### Attributes

| PARAMS |
|---|

**check_parameters**(*parameters*)

Verify that the parameters meet the operation specification.

**Parameters**
**parameters** (*dict*) – Dictionary of parameters for this operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

**do_op**(*dispatcher*, *df*, *name*, *sidecar=None*)

Factor the column using HED tag queries.

**Parameters**

- **dispatcher** (Dispatcher) – Manages the operation I/O.
- **df** (*DataFrame*) – The DataFrame to be remodeled.
- **name** (*str*) – Unique identifier for the dataframe – often the original file path.
- **sidecar** (Sidecar *or file-like*) – Only needed for HED operations.

**Returns**
A new dataframe after processing.

**Return type**
Dataframe

**Raises**
**ValueError** –

- If a name for a new query factor column is already a column.

## hed.tools.remodeling.operations.factor_hed_type_op

Create tabular file factors from type variables.

### Classes

| | |
|---|---|
| *FactorHedTypeOp*(parameters) | Create tabular file factors from type variables and append to tabular data. |

### hed.tools.remodeling.operations.factor_hed_type_op.FactorHedTypeOp

**class FactorHedTypeOp**(*parameters*)

Bases: *BaseOp*

Create tabular file factors from type variables and append to tabular data.

**Required remodeling parameters:**

- **type_tag** (*str*): HED tag used to find the factors (most commonly *condition-variable*).
- **type_values** (*list*): Factor values to include. If empty all values of that type_tag are used.

**__init__**(*parameters*)

Constructor for the factor HED type operation.

**Parameters**
**parameters** (*dict*) – Actual values of the parameters for the operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.
- **ValueError** –
  - If the specification is missing a valid operation.

#### Methods

| | |
|---|---|
| *__init__*(parameters) | Constructor for the factor HED type operation. |
| *check_parameters*(parameters) | Verify that the parameters meet the operation specification. |
| *do_op*(dispatcher, df, name[, sidecar]) | Factor columns based on HED type and append to tabular data. |

### Attributes

| PARAMS |
|---|

**check_parameters**(*parameters*)

Verify that the parameters meet the operation specification.

**Parameters**
**parameters** (*dict*) – Dictionary of parameters for this operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

**do_op**(*dispatcher*, *df*, *name*, *sidecar=None*)

Factor columns based on HED type and append to tabular data.

**Parameters**

- **dispatcher** (Dispatcher) – Manages the operation I/O.
- **df** (*DataFrame*) – The DataFrame to be remodeled.
- **name** (*str*) – Unique identifier for the dataframe – often the original file path.
- **sidecar** (Sidecar *or file-like*) – Only needed for HED operations.

**Returns**
A new DataFame with that includes the factors.

**Return type**
DataFrame

### Notes

- If column_name is not a column in df, df is just returned.

## hed.tools.remodeling.operations.merge_consecutive_op

Merge consecutive rows with same column value.

### Classes

| | |
|---|---|
| *MergeConsecutiveOp*(parameters) | Merge consecutive rows with same column value. |

### hed.tools.remodeling.operations.merge_consecutive_op.MergeConsecutiveOp

**class MergeConsecutiveOp**(*parameters*)

Bases: *BaseOp*

Merge consecutive rows with same column value.

**Required remodeling parameters:**

- **column_name** (*str*): name of column whose consecutive values are to be compared (the merge column).
- **event_code** (*str* or *int* or *float*): the particular value in the match column to be merged.
- **match_columns** (*list*): A list of columns whose values have to be matched for two events to be the same.
- **set_durations** (*bool*): If true, set the duration of the merged event to the extent of the merged events.
- **ignore_missing** (*bool*): If true, missing match_columns are ignored.

**__init__**(*parameters*)

Constructor for the merge consecutive operation.

**Parameters**
**parameters** (*dict*) – Actual values of the parameters for the operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.
- **ValueError** –
  - If the specification is missing a valid operation.
  - If one of the match column is the merge column.

### Methods

| | |
|---|---|
| *__init__*(parameters) | Constructor for the merge consecutive operation. |
| *check_parameters*(parameters) | Verify that the parameters meet the operation specification. |
| *do_op*(dispatcher, df, name[, sidecar]) | Merge consecutive rows with the same column value. |

### Attributes

| PARAMS |
|---|

**check_parameters**(*parameters*)

Verify that the parameters meet the operation specification.

**Parameters**
**parameters** (*dict*) – Dictionary of parameters for this operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

**do_op**(*dispatcher*, *df*, *name*, *sidecar=None*)

Merge consecutive rows with the same column value.

**Parameters**

- **dispatcher** (Dispatcher) – Manages the operation I/O.
- **df** (*DataFrame*) – The DataFrame to be remodeled.
- **name** (*str*) – Unique identifier for the dataframe – often the original file path.
- **sidecar** (Sidecar *or file-like*) – Not needed for this operation.

**Returns**
A new dataframe after processing.

**Return type**
Dataframe

**Raises**
**ValueError** –

- If dataframe does not have the anchor column and ignore_missing is False.
- If a match column is missing and ignore_missing is false.
- If the durations were to be set and the dataframe did not have an onset column.
- If the durations were to be set and the dataframe did not have a duration column.

### hed.tools.remodeling.operations.number_groups_op

Implementation in progress.

#### Classes

| | |
|---|---|
| *NumberGroupsOp*(parameters) | Implementation in progress. |

### hed.tools.remodeling.operations.number_groups_op.NumberGroupsOp

**class NumberGroupsOp**(*parameters*)

Bases: *BaseOp*

Implementation in progress.

**__init__**(*parameters*)

Base class constructor for operations.

**Parameters**

- **op_spec** (*dict*) – Specification for required and optional parameters.
- **parameters** (*dict*) – Actual values of the parameters for the operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.
- **ValueError** –
  - If the specification is missing a valid operation.

#### Methods

| | |
|---|---|
| *__init__*(parameters) | Base class constructor for operations. |
| *check_parameters*(parameters) | Verify that the parameters meet the operation specification. |
| *do_op*(dispatcher, df, name[, sidecar]) | Add numbers to groups of events in dataframe. |

### Attributes

| PARAMS |
|---|

**check_parameters**(*parameters*)

Verify that the parameters meet the operation specification.

**Parameters**
**parameters** (*dict*) – Dictionary of parameters for this operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

**do_op**(*dispatcher*, *df*, *name*, *sidecar=None*)

Add numbers to groups of events in dataframe.

**Parameters**

- **dispatcher** (Dispatcher) – Manages the operation I/O.
- **df** (*DataFrame*) – The DataFrame to be remodeled.
- **name** (*str*) – Unique identifier for the dataframe – often the original file path.
- **sidecar** (Sidecar *or file-like*) – Only needed for HED operations.

**Returns**
Dataframe - a new dataframe after processing.

## hed.tools.remodeling.operations.number_rows_op

Implementation in progress.

### Classes

| | |
|---|---|
| *NumberRowsOp*(parameters) | Implementation in progress. |

### hed.tools.remodeling.operations.number_rows_op.NumberRowsOp

**class NumberRowsOp**(*parameters*)

Bases: *BaseOp*

Implementation in progress.

**__init__**(*parameters*)

Base class constructor for operations.

**Parameters**

- **op_spec** (*dict*) – Specification for required and optional parameters.
- **parameters** (*dict*) – Actual values of the parameters for the operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.
- **ValueError** –
  - If the specification is missing a valid operation.

#### Methods

| | |
|---|---|
| *__init__*(parameters) | Base class constructor for operations. |
| *check_parameters*(parameters) | Verify that the parameters meet the operation specification. |
| *do_op*(dispatcher, df, name[, sidecar]) | Add numbers events dataframe. |

#### Attributes

| |
|---|
| PARAMS |

**check_parameters**(*parameters*)

Verify that the parameters meet the operation specification.

**Parameters**

**parameters** (*dict*) – Dictionary of parameters for this operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –

– If a parameter has the wrong type.

**do_op**(*dispatcher*, *df*, *name*, *sidecar=None*)

Add numbers events dataframe.

**Parameters**

- **dispatcher** (Dispatcher) – Manages operation I/O.
- **df** (*DataFrame*) –
  - The DataFrame to be remodeled.
- **name** (*str*) –
  - Unique identifier for the dataframe – often the original file path.
- **sidecar** (Sidecar *or file-like*) – Only needed for HED operations.

**Returns**

Dataframe - a new dataframe after processing.

### hed.tools.remodeling.operations.remap_columns_op

Map values in m columns into a new combinations in n columns.

### Classes

| | |
|---|---|
| *RemapColumnsOp*(parameters) | Map values in m columns into a new combinations in n columns. |

### hed.tools.remodeling.operations.remap_columns_op.RemapColumnsOp

**class RemapColumnsOp**(*parameters*)

Bases: *BaseOp*

Map values in m columns into a new combinations in n columns.

**Required remodeling parameters:**

- **source_columns** (*list*): The key columns to map (m key columns).
- **destination_columns** (*list*): The destination columns to have the mapped values (n destination columns).
- **map_list** (*list*): A list of lists with the mapping.
- **ignore_missing** (*bool*): If True, entries whose key column values are not in map_list are ignored.

**Optional remodeling parameters:**

**integer_sources** (*list*): Source columns that should be treated as integers rather than strings.

### Notes

Each list element list is of length m + n with the key columns followed by mapped columns.

TODO: Allow wildcards

**__init__**(*parameters*)

Constructor for the remap columns operation.

**Parameters**
**parameters** (*dict*) – Parameter values for required and optional parameters.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.
- **ValueError** –
  - If an integer column is not a key column.
  - If a column designated as an integer source does not have valid integers.
  - If no source columns are specified.
  - If no destination columns are specified.
  - If a map_list entry has the wrong number of items (source columns + destination columns).

### Methods

| | |
|---|---|
| *__init__*(parameters) | Constructor for the remap columns operation. |
| *check_parameters*(parameters) | Verify that the parameters meet the operation specification. |
| *do_op*(dispatcher, df, name[, sidecar]) | Remap new columns from combinations of others. |

### Attributes

| |
|---|
| PARAMS |

**check_parameters**(*parameters*)

Verify that the parameters meet the operation specification.

**Parameters**
**parameters** (*dict*) – Dictionary of parameters for this operation.

**Raises**

- **KeyError** –

- If a required parameter is missing.
- If an unexpected parameter is provided.

- **TypeError** –
  - If a parameter has the wrong type.

**do_op**(*dispatcher*, *df*, *name*, *sidecar=None*)

Remap new columns from combinations of others.

**Parameters**

- **dispatcher** (Dispatcher) – Manages the operation I/O.
- **df** (*DataFrame*) – The DataFrame to be remodeled.
- **name** (*str*) – Unique identifier for the dataframe – often the original file path.
- **sidecar** (Sidecar *or file-like*) – Not needed for this operation.

**Returns**
A new dataframe after processing.

**Return type**
Dataframe

**Raises**
**ValueError** –

- If ignore_missing is false and source values from the data are not in the map.

### hed.tools.remodeling.operations.remove_columns_op

Remove columns from a tabular file.

### Classes

| | |
|---|---|
| *RemoveColumnsOp*(parameters) | Remove columns from a tabular file. |

### hed.tools.remodeling.operations.remove_columns_op.RemoveColumnsOp

**class RemoveColumnsOp**(*parameters*)

Bases: *BaseOp*

Remove columns from a tabular file.

**Required remodeling parameters:**

- **remove_names** (*list*): The names of the columns to be removed.
- **ignore_missing** (*boolean*): If true, names in remove_names that are not columns in df should be ignored.

**__init__**(*parameters*)

Constructor for remove columns operation.

**Parameters**
**parameters** (*dict*) – Dictionary with the parameter values for required and optional parameters

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

### Methods

| | |
|---|---|
| *__init__*(parameters) | Constructor for remove columns operation. |
| *check_parameters*(parameters) | Verify that the parameters meet the operation specification. |
| *do_op*(dispatcher, df, name[, sidecar]) | Remove indicated columns from a dataframe. |

### Attributes

| |
|---|
| PARAMS |

**check_parameters**(*parameters*)

Verify that the parameters meet the operation specification.

**Parameters**
**parameters** (*dict*) – Dictionary of parameters for this operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

**do_op**(*dispatcher*, *df*, *name*, *sidecar=None*)

Remove indicated columns from a dataframe.

**Parameters**

- **dispatcher** (*Dispatcher*) – Manages the operation I/O.
- **df** (*DataFrame*) – The DataFrame to be remodeled.
- **name** (*str*) – Unique identifier for the dataframe – often the original file path.
- **sidecar** (*Sidecar or file-like*) – Not needed for this operation.

**Returns**
A new dataframe after processing.

**Return type**
Dataframe

**Raises**
**KeyError** –

- If ignore_missing is False and a column not in the data is to be removed.

### hed.tools.remodeling.operations.remove_rows_op

Remove rows from a tabular file.

### Classes

| | |
|---|---|
| *RemoveRowsOp*(parameters) | Remove rows from a tabular file. |

### hed.tools.remodeling.operations.remove_rows_op.RemoveRowsOp

**class RemoveRowsOp**(*parameters*)

Bases: *BaseOp*

Remove rows from a tabular file.

**Required remodeling parameters:**

- **column_name** (*str*): The name of column to be tested.
- **remove_values** (*list*): The values to test for row removal.

**__init__**(*parameters*)

Constructor for remove rows operation.

**Parameters**
**parameters** (*dict*) – Dictionary with the parameter values for required and optional parameters.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

### Methods

| | |
|---|---|
| *__init__*(parameters) | Constructor for remove rows operation. |
| *check_parameters*(parameters) | Verify that the parameters meet the operation specification. |
| *do_op*(dispatcher, df, name[, sidecar]) | Remove rows with the values indicated in the column. |

### Attributes

| |
|---|
| PARAMS |

**check_parameters**(*parameters*)

Verify that the parameters meet the operation specification.

**Parameters**
**parameters** (*dict*) – Dictionary of parameters for this operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

**do_op**(*dispatcher*, *df*, *name*, *sidecar=None*)

Remove rows with the values indicated in the column.

**Parameters**

- **dispatcher** (Dispatcher) – Manages the operation I/O.
- **df** (*DataFrame*) – The DataFrame to be remodeled.
- **name** (*str*) – Unique identifier for the dataframe – often the original file path.
- **sidecar** (Sidecar *or file-like*) – Not needed for this operation.

**Returns**
A new dataframe after processing.

**Return type**
Dataframe

### hed.tools.remodeling.operations.rename_columns_op

Rename columns in a tabular file.

### Classes

| | |
|---|---|
| *RenameColumnsOp*(parameters) | Rename columns in a tabular file. |

### hed.tools.remodeling.operations.rename_columns_op.RenameColumnsOp

**class RenameColumnsOp**(*parameters*)

Bases: *BaseOp*

Rename columns in a tabular file.

**Required remodeling parameters:**

- **column_mapping** (*dict*): The names of the columns to be removed.
- **ignore_missing** (*bool*): If true, the names in remove_names that are not columns and should be ignored.

**__init__**(*parameters*)

Constructor for rename columns operation.

**Parameters**

**parameters** (*dict*) – Dictionary with the parameter values for required and optional parameters

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

### Methods

| | |
|---|---|
| *__init__*(parameters) | Constructor for rename columns operation. |
| *check_parameters*(parameters) | Verify that the parameters meet the operation specification. |
| *do_op*(dispatcher, df, name[, sidecar]) | Rename columns as specified in column_mapping dictionary. |

### Attributes

| PARAMS |
| --- |

**check_parameters**(*parameters*)

Verify that the parameters meet the operation specification.

**Parameters**
**parameters** (*dict*) – Dictionary of parameters for this operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

**do_op**(*dispatcher*, *df*, *name*, *sidecar=None*)

Rename columns as specified in column_mapping dictionary.

**Parameters**

- **dispatcher** (Dispatcher) – Manages the operation I/O.
- **df** (*DataFrame*) – The DataFrame to be remodeled.
- **name** (*str*) – Unique identifier for the dataframe – often the original file path.
- **sidecar** (Sidecar *or file-like*) – Not needed for this operation.

**Returns**
A new dataframe after processing.

**Return type**
Dataframe

**Raises**
**KeyError** –

- When ignore_missing is false and column_mapping has columns not in the data.

## hed.tools.remodeling.operations.reorder_columns_op

Reorder columns in a tabular file.

### Classes

| | |
|---|---|
| *ReorderColumnsOp*(parameters) | Reorder columns in a tabular file. |

### hed.tools.remodeling.operations.reorder_columns_op.ReorderColumnsOp

**class ReorderColumnsOp**(*parameters*)

Bases: *BaseOp*

Reorder columns in a tabular file.

**Required parameters:**

- column_order (*list*): The names of the columns to be reordered.
- ignore_missing (*bool*): If false and a column in column_order is not in df, skip the column
- keep_others (*bool*): If true, columns not in column_order are placed at end.

**__init__**(*parameters*)

Constructor for reorder columns operation.

**Parameters**
**parameters** (*dict*) – Dictionary with the parameter values for required and optional parameters.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

#### Methods

| | |
|---|---|
| *__init__*(parameters) | Constructor for reorder columns operation. |
| *check_parameters*(parameters) | Verify that the parameters meet the operation specification. |
| *do_op*(dispatcher, df, name[, sidecar]) | Reorder columns as specified in event dictionary. |

#### Attributes

| |
|---|
| PARAMS |

**check_parameters**(*parameters*)

Verify that the parameters meet the operation specification.

**Parameters**
**parameters** (*dict*) – Dictionary of parameters for this operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

**do_op**(*dispatcher*, *df*, *name*, *sidecar=None*)

Reorder columns as specified in event dictionary.

**Parameters**

- **dispatcher** (Dispatcher) – Manages the operation I/O.
- **df** (*DataFrame*) – The DataFrame to be remodeled.
- **name** (*str*) – Unique identifier for the dataframe – often the original file path.
- **sidecar** (Sidecar *or file-like*) – Not needed for this operation.

**Returns**
A new dataframe after processing.

**Return type**
Dataframe

**Raises**
**ValueError** –

- When ignore_missing is false and column_order has columns not in the data.

### hed.tools.remodeling.operations.split_rows_op

Split rows in a tabular file into multiple rows based on a column.

### Classes

| | |
|---|---|
| *SplitRowsOp*(parameters) | Split rows in a tabular file into multiple rows based on parameters. |

### hed.tools.remodeling.operations.split_rows_op.SplitRowsOp

**class SplitRowsOp**(*parameters*)

Bases: *BaseOp*

Split rows in a tabular file into multiple rows based on parameters.

**Required remodeling parameters:**

- **anchor_column** (*str*): The column in which the names of new items are stored.
- **new_events** (*dict*): Mapping of new values based on values in the original row.
- **remove_parent_row** (*bool*): If true, the original row that was split is removed.

**__init__**(*parameters*)

Constructor for the split rows operation.

**Parameters**

**parameters** (*dict*) – Dictionary with the parameter values for required and optional parameters.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

### Methods

| | |
|---|---|
| *__init__*(parameters) | Constructor for the split rows operation. |
| *check_parameters*(parameters) | Verify that the parameters meet the operation specification. |
| *do_op*(dispatcher, df, name[, sidecar]) | Split a row representing a particular event into multiple rows. |

### Attributes

| |
|---|
| PARAMS |

**check_parameters**(*parameters*)

Verify that the parameters meet the operation specification.

**Parameters**

**parameters** (*dict*) – Dictionary of parameters for this operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

**do_op**(*dispatcher*, *df*, *name*, *sidecar=None*)

Split a row representing a particular event into multiple rows.

**Parameters**

- **dispatcher** (Dispatcher) – Manages the operation I/O.
- **df** (*DataFrame*) – The DataFrame to be remodeled.

- **name** (*str*) – Unique identifier for the dataframe – often the original file path.
- **sidecar** (Sidecar *or file-like*) – Not needed for this operation.

**Returns**
A new dataframe after processing.

**Return type**
Dataframe

**Raises**
**TypeError** – -If bad onset or duration.

### hed.tools.remodeling.operations.summarize_column_names_op

Summarize the column names in a collection of tabular files.

### Classes

| | |
|---|---|
| *ColumnNamesSummary*(sum_op) | |
| *SummarizeColumnNamesOp*(parameters) | Summarize the column names in a collection of tabular files. |

### hed.tools.remodeling.operations.summarize_column_names_op.ColumnNamesSummary

**class ColumnNamesSummary**(*sum_op*)

Bases: *BaseSummary*

**__init__**(*sum_op*)

### Methods

| | |
|---|---|
| *__init__*(sum_op) | |
| dump_summary(filename, summary) | |
| *get_details_dict*(column_summary) | Return the summary dictionary extracted from a ColumnNameSummary. |
| get_individual(summary_details[, separately]) | |
| *get_summary*([individual_summaries]) | Return a summary dictionary with the information. |
| *get_summary_details*([include_individual]) | Return a dictionary with the details for individual files and the overall dataset. |
| get_text_summary([individual_summaries]) | |
| get_text_summary_details([include_individual]) | |
| *merge_all_info*() | Create a ColumnNameSummary containing the overall dataset summary. |
| save(save_dir[, file_formats, ...]) | |
| *update_summary*(new_info) | Update the summary for a given tabular input file. |

### Attributes

| |
|---|
| DISPLAY_INDENT |
| INDIVIDUAL_SUMMARIES_PATH |

**get_details_dict**(*column_summary*)

Return the summary dictionary extracted from a ColumnNameSummary.

**Parameters**
**column_summary** (ColumnNameSummary) – A column name summary for the data file.

**Returns**
dict - a dictionary with the summary information for column names.

**get_summary**(*individual_summaries='separate'*)

Return a summary dictionary with the information.

**Parameters**
**individual_summaries** (*str*) – "separate", "consolidated", or "none"

**Returns**
dict - dictionary with "Dataset" and "Individual files" keys.

**Notes: The individual_summaries value is processed as follows**

- "separate" individual summaries are to be in separate files
- "consolidated" means that the individual summaries are in same file as overall summary

- "none" means that only the overall summary is produced.

`get_summary_details`(*include_individual=True*)

Return a dictionary with the details for individual files and the overall dataset.

**Parameters**
**include_individual** (*bool*) – If True, summaries for individual files are included.

**Returns**
dict - a dictionary with 'Dataset' and 'Individual files' keys.

### Notes

- The 'Dataset' value is either a string or a dictionary with the overall summary.
- **The 'Individual files' value is dictionary whose keys are file names and values are** their corresponding summaries.

Users are expected to provide merge_all_info and get_details_dict to support this.

`merge_all_info`()

Create a ColumnNameSummary containing the overall dataset summary.

**Returns**
ColumnNameSummary - the overall summary object for column names.

`update_summary`(*new_info*)

Update the summary for a given tabular input file.

**Parameters**
**new_info** (*dict*) – A dictionary with the parameters needed to update a summary.

### Notes

- The summary information is kept in separate ColumnNameSummary objects for each file.
- The summary needs a "name" str and a "column_names" list.
- The summary uses ColumnNameSummary as the summary object.

## hed.tools.remodeling.operations.summarize_column_names_op.SummarizeColumnNamesOp

`class SummarizeColumnNamesOp`(*parameters*)

Bases: *BaseOp*

Summarize the column names in a collection of tabular files.

**Required remodeling parameters:**

- **summary_name** (*str*) The name of the summary.
- **summary_filename** (*str*) Base filename of the summary.

The purpose is to check that all the tabular files have the same columns in same order.

**__init__**(*parameters*)

Constructor for summarize column names operation.

**Parameters**

**parameters** (*dict*) – Dictionary with the parameter values for required and optional parameters.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

### Methods

| | |
|---|---|
| *__init__*(parameters) | Constructor for summarize column names operation. |
| *check_parameters*(parameters) | Verify that the parameters meet the operation specification. |
| *do_op*(dispatcher, df, name[, sidecar]) | Create a column name summary for df. |

### Attributes

| |
|---|
| PARAMS |
| SUMMARY_TYPE |

**check_parameters**(*parameters*)

Verify that the parameters meet the operation specification.

**Parameters**

**parameters** (*dict*) – Dictionary of parameters for this operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

**do_op**(*dispatcher*, *df*, *name*, *sidecar=None*)

Create a column name summary for df.

**Parameters**

- **dispatcher** (*Dispatcher*) – Manages the operation I/O.
- **df** (*DataFrame*) – The DataFrame to be remodeled.

- **name** (*str*) – Unique identifier for the dataframe – often the original file path.
- **sidecar** (Sidecar *or file-like*) – Not needed for this operation.

**Returns**
A copy of df.

**Return type**
DataFrame

**Side-effect:**
Updates the relevant summary.

### hed.tools.remodeling.operations.summarize_column_values_op

Summarize the values in the columns of a tabular file.

### Classes

| | |
|---|---|
| *ColumnValueSummary*(sum_op) | |
| *SummarizeColumnValuesOp*(parameters) | Summarize the values in the columns of a tabular file. |

### hed.tools.remodeling.operations.summarize_column_values_op.ColumnValueSummary

**class ColumnValueSummary**(*sum_op*)

Bases: *BaseSummary*

**__init__**(*sum_op*)

### Methods

| | |
|---|---|
| *__init__*(sum_op) | |
| dump_summary(filename, summary) | |
| *get_details_dict*(summary) | Return a dictionary with the summary contained in a TabularSummary |
| get_individual(summary_details[, separately]) | |
| get_list_str(lst) | |
| *get_summary*([individual_summaries]) | Return a summary dictionary with the information. |
| *get_summary_details*([include_individual]) | Return a dictionary with the details for individual files and the overall dataset. |
| get_text_summary([individual_summaries]) | |
| get_text_summary_details([include_individual]) | |
| *merge_all_info*() | Create a TabularSummary containing the overall dataset summary. |
| *partition_list*(lst, n) | Partition a list into lists of n items. |
| save(save_dir[, file_formats, ...]) | |
| sort_dict(count_dict[, reverse]) | |
| *update_summary*(new_info) | Update the summary for a given tabular input file. |

### Attributes

| |
|---|
| DISPLAY_INDENT |
| INDIVIDUAL_SUMMARIES_PATH |

**get_details_dict**(*summary*)

Return a dictionary with the summary contained in a TabularSummary

**Parameters**
**summary** (TabularSummary) – Dictionary of merged summary information.

**Returns**
Dictionary with the information suitable for extracting printout.

**Return type**
dict

**get_summary**(*individual_summaries='separate'*)

Return a summary dictionary with the information.

**Parameters**
**individual_summaries** (*str*) – "separate", "consolidated", or "none"

**Returns**
dict - dictionary with "Dataset" and "Individual files" keys.

**Notes: The individual_summaries value is processed as follows**

- "separate" individual summaries are to be in separate files
- "consolidated" means that the individual summaries are in same file as overall summary
- "none" means that only the overall summary is produced.

`get_summary_details`(*include_individual=True*)

Return a dictionary with the details for individual files and the overall dataset.

**Parameters**
**include_individual** (*bool*) – If True, summaries for individual files are included.

**Returns**
dict - a dictionary with 'Dataset' and 'Individual files' keys.

### Notes

- The 'Dataset' value is either a string or a dictionary with the overall summary.
- **The 'Individual files' value is dictionary whose keys are file names and values are** their corresponding summaries.

Users are expected to provide merge_all_info and get_details_dict to support this.

`merge_all_info`()

Create a TabularSummary containing the overall dataset summary.

**Returns**
TabularSummary - the summary object for column values.

`static partition_list`(*lst*, *n*)

Partition a list into lists of n items.

**Parameters**

- **lst** (*list*) – List to be partitioned
- **n** (*int*) – Number of items in each sublist

**Returns**
list of lists of n elements, the last might have fewer.

**Return type**
list

`update_summary`(*new_info*)

Update the summary for a given tabular input file.

**Parameters**
**new_info** (*dict*) – A dictionary with the parameters needed to update a summary.

### Notes

- The summary information is kept in separate TabularSummary objects for each file.
- The summary needs a "name" str and a "df" .

## hed.tools.remodeling.operations.summarize_column_values_op.SummarizeColumnValuesOp

**class SummarizeColumnValuesOp**(*parameters*)

Bases: *BaseOp*

Summarize the values in the columns of a tabular file.

**Required remodeling parameters:**

- **summary_name** (*str*): The name of the summary.
- **summary_filename** (*str*): Base filename of the summary.
- **skip_columns** (*list*): Names of columns to skip in the summary.
- **value_columns** (*list*): Names of columns to treat as value columns rather than categorical columns.

**Optional remodeling parameters:**

- **max_categorical** (*int*): Maximum number of unique values to include in summary for a categorical column.

The purpose is to produce a summary of the values in a tabular file.

**__init__**(*parameters*)

Constructor for the summarize column values operation.

**Parameters**
**parameters** (*dict*) – Dictionary with the parameter values for required and optional parameters.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

### Methods

| | |
|---|---|
| *__init__*(parameters) | Constructor for the summarize column values operation. |
| *check_parameters*(parameters) | Verify that the parameters meet the operation specification. |
| *do_op*(dispatcher, df, name[, sidecar]) | Create a summary of the column values in df. |

### Attributes

| |
|---|
| MAX_CATEGORICAL |
| PARAMS |
| SUMMARY_TYPE |
| VALUES_PER_LINE |

**check_parameters**(*parameters*)

Verify that the parameters meet the operation specification.

**Parameters**
**parameters** (*dict*) – Dictionary of parameters for this operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

**do_op**(*dispatcher*, *df*, *name*, *sidecar=None*)

Create a summary of the column values in df.

**Parameters**

- **dispatcher** (Dispatcher) – Manages the operation I/O.
- **df** (*DataFrame*) – The DataFrame to be remodeled.
- **name** (*str*) – Unique identifier for the dataframe – often the original file path.
- **sidecar** (Sidecar *or file-like*) – Not needed for this operation.

**Returns**
A copy of df.

**Return type**
DataFrame

**Side-effect:**
Updates the relevant summary.

### hed.tools.remodeling.operations.summarize_definitions_op

Summarize the type_defs in the dataset.

### Classes

| | |
|---|---|
| *DefinitionSummary*(sum_op, hed_schema[, ...]) | |
| *SummarizeDefinitionsOp*(parameters) | Summarize the type_defs in the dataset. |

### hed.tools.remodeling.operations.summarize_definitions_op.DefinitionSummary

**class DefinitionSummary**(*sum_op*, *hed_schema*, *known_defs=None*)

Bases: *BaseSummary*

**__init__**(*sum_op*, *hed_schema*, *known_defs=None*)

#### Methods

| | |
|---|---|
| *__init__*(sum_op, hed_schema[, known_defs]) | |
| dump_summary(filename, summary) | |
| *get_details_dict*(def_gatherer) | Return the summary-specific information in a dictionary. |
| get_individual(summary_details[, separately]) | |
| *get_summary*([individual_summaries]) | Return a summary dictionary with the information. |
| *get_summary_details*([include_individual]) | Return a dictionary with the details for individual files and the overall dataset. |
| get_text_summary([individual_summaries]) | |
| get_text_summary_details([include_individual]) | |
| *merge_all_info*() | Create an Object containing the definition summary. |
| save(save_dir[, file_formats, ...]) | |
| *update_summary*(new_info) | Update the summary for a given tabular input file. |

### Attributes

| |
|---|
| DISPLAY_INDENT |
| INDIVIDUAL_SUMMARIES_PATH |

**get_details_dict**(*def_gatherer*)

Return the summary-specific information in a dictionary.

**Parameters**
**def_gatherer** (DefExpandGatherer) – Contains the resolved dictionaries.

**Returns**
dictionary with the summary results.

**Return type**
dict

**get_summary**(*individual_summaries='separate'*)

Return a summary dictionary with the information.

**Parameters**
**individual_summaries** (*str*) – "separate", "consolidated", or "none"

**Returns**
dict - dictionary with "Dataset" and "Individual files" keys.

**Notes: The individual_summaries value is processed as follows**

- "separate" individual summaries are to be in separate files
- "consolidated" means that the individual summaries are in same file as overall summary
- "none" means that only the overall summary is produced.

**get_summary_details**(*include_individual=True*)

Return a dictionary with the details for individual files and the overall dataset.

**Parameters**
**include_individual** (*bool*) – If True, summaries for individual files are included.

**Returns**
dict - a dictionary with 'Dataset' and 'Individual files' keys.

### Notes

- The 'Dataset' value is either a string or a dictionary with the overall summary.
- **The 'Individual files' value is dictionary whose keys are file names and values are** their corresponding summaries.

Users are expected to provide merge_all_info and get_details_dict to support this.

**merge_all_info**()

Create an Object containing the definition summary.

**Returns**
Object - the overall summary object for type_defs.

`update_summary`(*new_info*)

Update the summary for a given tabular input file.

**Parameters**
**new_info** (*dict*) – A dictionary with the parameters needed to update a summary.

### Notes

- The summary needs a "name" str, a "schema" and a "Sidecar".

## hed.tools.remodeling.operations.summarize_definitions_op.SummarizeDefinitionsOp

`class SummarizeDefinitionsOp`(*parameters*)

Bases: *BaseOp*

Summarize the type_defs in the dataset.

**Required remodeling parameters:**

- **summary_name** (*str*): The name of the summary.
- **summary_filename** (*str*): Base filename of the summary.

The purpose is to produce a summary of the values in a tabular file.

`__init__`(*parameters*)

Constructor for the summarize column values operation.

**Parameters**
**parameters** (*dict*) – Dictionary with the parameter values for required and optional parameters.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

### Methods

| | |
|---|---|
| *__init__*(parameters) | Constructor for the summarize column values operation. |
| *check_parameters*(parameters) | Verify that the parameters meet the operation specification. |
| *do_op*(dispatcher, df, name[, sidecar]) | Create summaries of type_defs |

### Attributes

| |
|---|
| PARAMS |
| SUMMARY_TYPE |

**check_parameters**(*parameters*)

Verify that the parameters meet the operation specification.

**Parameters**
**parameters** (*dict*) – Dictionary of parameters for this operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

**do_op**(*dispatcher*, *df*, *name*, *sidecar=None*)

Create summaries of type_defs

**Parameters**

- **dispatcher** (Dispatcher) – Manages the operation I/O.
- **df** (*DataFrame*) – The DataFrame to be remodeled.
- **name** (*str*) – Unique identifier for the dataframe – often the original file path.
- **sidecar** (Sidecar *or file-like*) – Only needed for HED operations.

**Returns**
a copy of df

**Return type**
DataFrame

**Side-effect:**
Updates the relevant summary.

## hed.tools.remodeling.operations.summarize_hed_tags_op

Summarize the HED tags in collection of tabular files.

### Classes

| | |
|---|---|
| *HedTagSummary*(sum_op) | |
| *SummarizeHedTagsOp*(parameters) | Summarize the HED tags in collection of tabular files. |

### hed.tools.remodeling.operations.summarize_hed_tags_op.HedTagSummary

**class HedTagSummary**(*sum_op*)

Bases: *BaseSummary*

**__init__**(*sum_op*)

#### Methods

| | |
|---|---|
| *__init__*(sum_op) | |
| dump_summary(filename, summary) | |
| *get_details_dict*(tag_counts) | Return the summary-specific information in a dictionary. |
| get_individual(summary_details[, separately]) | |
| *get_summary*([individual_summaries]) | Return a summary dictionary with the information. |
| *get_summary_details*([include_individual]) | Return a dictionary with the details for individual files and the overall dataset. |
| get_text_summary([individual_summaries]) | |
| get_text_summary_details([include_individual]) | |
| *merge_all_info*() | Create a HedTagCounts containing the overall dataset HED tag summary. |
| save(save_dir[, file_formats, ...]) | |
| *update_summary*(new_info) | Update the summary for a given tabular input file. |

#### Attributes

| |
|---|
| DISPLAY_INDENT |
| INDIVIDUAL_SUMMARIES_PATH |

**get_details_dict**(*tag_counts*)

Return the summary-specific information in a dictionary.

**Parameters**

**tag_counts** (HedTagCounts) – Contains the counts of tags in the dataset.

**Returns**
dictionary with the summary results.

**Return type**
dict

`get_summary`(*individual_summaries='separate'*)

Return a summary dictionary with the information.

**Parameters**
**individual_summaries** (*str*) – "separate", "consolidated", or "none"

**Returns**
dict - dictionary with "Dataset" and "Individual files" keys.

**Notes: The individual_summaries value is processed as follows**

- "separate" individual summaries are to be in separate files
- "consolidated" means that the individual summaries are in same file as overall summary
- "none" means that only the overall summary is produced.

`get_summary_details`(*include_individual=True*)

Return a dictionary with the details for individual files and the overall dataset.

**Parameters**
**include_individual** (*bool*) – If True, summaries for individual files are included.

**Returns**
dict - a dictionary with 'Dataset' and 'Individual files' keys.

### Notes

- The 'Dataset' value is either a string or a dictionary with the overall summary.
- **The 'Individual files' value is dictionary whose keys are file names and values are** their corresponding summaries.

Users are expected to provide merge_all_info and get_details_dict to support this.

`merge_all_info`()

Create a HedTagCounts containing the overall dataset HED tag summary.

**Returns**
HedTagCounts - the overall dataset summary object for HED tag counts.

`update_summary`(*new_info*)

Update the summary for a given tabular input file.

**Parameters**
**new_info** (*dict*) – A dictionary with the parameters needed to update a summary.

### Notes

- The summary needs a "name" str, a "schema", a "df, and a "Sidecar".

## hed.tools.remodeling.operations.summarize_hed_tags_op.SummarizeHedTagsOp

**class SummarizeHedTagsOp**(*parameters*)

Bases: *BaseOp*

Summarize the HED tags in collection of tabular files.

**Required remodeling parameters:**

- **summary_name** (*str*): The name of the summary.
- **summary_filename** (*str*): Base filename of the summary.
- **tags** (*dict*): Specifies how to organize the tag output.

**Optional remodeling parameters:**

- **expand_context** (*bool*): If True, include counts from expanded context (not supported).

The purpose of this op is to produce a summary of the occurrences of hed tags organized in a specified manner. The

**__init__**(*parameters*)

Constructor for the summarize_hed_tags operation.

**Parameters**
**parameters** (*dict*) – Dictionary with the parameter values for required and optional parameters.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

### Methods

| | |
|---|---|
| *__init__*(parameters) | Constructor for the summarize_hed_tags operation. |
| *check_parameters*(parameters) | Verify that the parameters meet the operation specification. |
| *do_op*(dispatcher, df, name[, sidecar]) | Summarize the HED tags present in the dataset. |

### Attributes

| |
|---|
| PARAMS |
| SUMMARY_TYPE |

**check_parameters**(*parameters*)

Verify that the parameters meet the operation specification.

**Parameters**

**parameters** (*dict*) – Dictionary of parameters for this operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

**do_op**(*dispatcher*, *df*, *name*, *sidecar=None*)

Summarize the HED tags present in the dataset.

**Parameters**

- **dispatcher** (Dispatcher) – Manages the operation I/O.
- **df** (*DataFrame*) – The DataFrame to be remodeled.
- **name** (*str*) – Unique identifier for the dataframe – often the original file path.
- **sidecar** (Sidecar *or file-like*) – Only needed for HED operations.

**Returns**

A copy of df.

**Return type**

DataFrame

**Side effect:**

Updates the context.

## hed.tools.remodeling.operations.summarize_hed_type_op

Summarize a HED type tag in a collection of tabular files.

### Classes

| | |
|---|---|
| *HedTypeSummary*(sum_op) | |
| *SummarizeHedTypeOp*(parameters) | Summarize a HED type tag in a collection of tabular files. |

## hed.tools.remodeling.operations.summarize_hed_type_op.HedTypeSummary

**class HedTypeSummary**(*sum_op*)

Bases: *BaseSummary*

**__init__**(*sum_op*)

### Methods

| | |
|---|---|
| *__init__*(sum_op) | |
| dump_summary(filename, summary) | |
| *get_details_dict*(counts) | Return the summary-specific information in a dictionary. |
| get_individual(summary_details[, separately]) | |
| *get_summary*([individual_summaries]) | Return a summary dictionary with the information. |
| *get_summary_details*([include_individual]) | Return a dictionary with the details for individual files and the overall dataset. |
| get_text_summary([individual_summaries]) | |
| get_text_summary_details([include_individual]) | |
| *merge_all_info*() | Create a HedTypeCounts containing the overall dataset HED type summary. |
| save(save_dir[, file_formats, ...]) | |
| *update_summary*(new_info) | Update the summary for a given tabular input file. |

### Attributes

| |
|---|
| DISPLAY_INDENT |
| INDIVIDUAL_SUMMARIES_PATH |

**get_details_dict**(*counts*)

Return the summary-specific information in a dictionary.

**Parameters**

**counts** (HedTypeCounts) – Contains the counts of the events in which the type occurs.

**Returns**
dictionary with the summary results.

**Return type**
dict

`get_summary`(*individual_summaries='separate'*)

Return a summary dictionary with the information.

**Parameters**
**individual_summaries** (*str*) – "separate", "consolidated", or "none"

**Returns**
dict - dictionary with "Dataset" and "Individual files" keys.

**Notes: The individual_summaries value is processed as follows**

- "separate" individual summaries are to be in separate files
- "consolidated" means that the individual summaries are in same file as overall summary
- "none" means that only the overall summary is produced.

`get_summary_details`(*include_individual=True*)

Return a dictionary with the details for individual files and the overall dataset.

**Parameters**
**include_individual** (*bool*) – If True, summaries for individual files are included.

**Returns**
dict - a dictionary with 'Dataset' and 'Individual files' keys.

### Notes

- The 'Dataset' value is either a string or a dictionary with the overall summary.
- **The 'Individual files' value is dictionary whose keys are file names and values are** their corresponding summaries.

Users are expected to provide merge_all_info and get_details_dict to support this.

`merge_all_info`()

Create a HedTypeCounts containing the overall dataset HED type summary.

**Returns**
HedTypeCounts - the overall dataset summary object for HED type summary.

`update_summary`(*new_info*)

Update the summary for a given tabular input file.

**Parameters**
**new_info** (*dict*) – A dictionary with the parameters needed to update a summary.

### Notes

- The summary needs a "name" str, a "schema", a "df, and a "Sidecar".

## hed.tools.remodeling.operations.summarize_hed_type_op.SummarizeHedTypeOp

**class SummarizeHedTypeOp**(*parameters*)

Bases: *BaseOp*

Summarize a HED type tag in a collection of tabular files.

**Required remodeling parameters:**

- **summary_name** (*str*): The name of the summary.
- **summary_filename** (*str*): Base filename of the summary.
- **type_tag** (*str*):Type tag to get_summary (e.g. *condition-variable* or *task* tags).

The purpose of this op is to produce a summary of the occurrences of specified tag. This summary is often used with *condition-variable* to produce a summary of the experimental design.

**__init__**(*parameters*)

Constructor for the summarize hed type operation.

**Parameters**

**parameters** (*dict*) – Dictionary with the parameter values for required and optional parameters.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

### Methods

| | |
|---|---|
| *__init__*(parameters) | Constructor for the summarize hed type operation. |
| *check_parameters*(parameters) | Verify that the parameters meet the operation specification. |
| *do_op*(dispatcher, df, name[, sidecar]) | Summarize a specified HED type variable such as Condition-variable . |

### Attributes

| |
|---|
| PARAMS |
| SUMMARY_TYPE |

**check_parameters**(*parameters*)

Verify that the parameters meet the operation specification.

**Parameters**
**parameters** (*dict*) – Dictionary of parameters for this operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

**do_op**(*dispatcher*, *df*, *name*, *sidecar=None*)

Summarize a specified HED type variable such as Condition-variable .

**Parameters**

- **dispatcher** (Dispatcher) – Manages the operation I/O.
- **df** (*DataFrame*) – The DataFrame to be summarized.
- **name** (*str*) – Unique identifier for the dataframe – often the original file path.
- **sidecar** (Sidecar *or file-like*) – Usually required unless event file has a HED column.

**Returns**
A copy of df

**Return type**
DataFrame

**Side effect:**
Updates the relevant summary.

## hed.tools.remodeling.operations.summarize_hed_validation_op

Validate the HED tags in a dataset and report errors.

### Classes

| | |
|---|---|
| *HedValidationSummary*(sum_op) | |
| *SummarizeHedValidationOp*(parameters) | Validate the HED tags in a dataset and report errors. |

### hed.tools.remodeling.operations.summarize_hed_validation_op.HedValidationSummary

**class HedValidationSummary**(*sum_op*)

Bases: *BaseSummary*

**__init__**(*sum_op*)

#### Methods

| | |
|---|---|
| *__init__*(sum_op) | |
| dump_summary(filename, summary) | |
| *get_details_dict*(summary_info) | Return the summary details from the summary_info. |
| get_empty_results() | |
| get_error_list(error_dict[, count_only, indent]) | |
| get_individual(summary_details[, separately]) | |
| *get_summary*([individual_summaries]) | Return a summary dictionary with the information. |
| *get_summary_details*([include_individual]) | Return a dictionary with the details for individual files and the overall dataset. |
| get_text_summary([individual_summaries]) | |
| get_text_summary_details([include_individual]) | |
| *merge_all_info*() | Create a dictionary containing all the errors in the dataset. |
| save(save_dir[, file_formats, ...]) | |
| update_error_location(error_locations, ...) | |
| *update_summary*(new_info) | Update the summary for a given tabular input file. |

### Attributes

| DISPLAY_INDENT |
| --- |
| INDIVIDUAL_SUMMARIES_PATH |

**get_details_dict**(*summary_info*)

Return the summary details from the summary_info.

**Parameters**
**summary_info** (*dict*) – Dictionary of issues

**Returns**
Same summary_info as was passed in.

**Return type**
dict

**get_summary**(*individual_summaries='separate'*)

Return a summary dictionary with the information.

**Parameters**
**individual_summaries** (*str*) – "separate", "consolidated", or "none"

**Returns**
dict - dictionary with "Dataset" and "Individual files" keys.

**Notes: The individual_summaries value is processed as follows**

- "separate" individual summaries are to be in separate files
- "consolidated" means that the individual summaries are in same file as overall summary
- "none" means that only the overall summary is produced.

**get_summary_details**(*include_individual=True*)

Return a dictionary with the details for individual files and the overall dataset.

**Parameters**
**include_individual** (*bool*) – If True, summaries for individual files are included.

**Returns**
dict - a dictionary with 'Dataset' and 'Individual files' keys.

### Notes

- The 'Dataset' value is either a string or a dictionary with the overall summary.
- **The 'Individual files' value is dictionary whose keys are file names and values are** their corresponding summaries.

Users are expected to provide merge_all_info and get_details_dict to support this.

**merge_all_info**()

Create a dictionary containing all the errors in the dataset.

**Returns**
dict - dictionary of issues organized into sidecar_issues and event_issues.

`update_summary`(*new_info*)

Update the summary for a given tabular input file.

**Parameters**
**new_info** (*dict*) – A dictionary with the parameters needed to update a summary.

### Notes

- The summary needs a "name" str, a schema, a "df", and a "Sidecar".

## hed.tools.remodeling.operations.summarize_hed_validation_op.SummarizeHedValidationOp

`class SummarizeHedValidationOp`(*parameters*)

Bases: *BaseOp*

Validate the HED tags in a dataset and report errors.

**Required remodeling parameters:**

- **summary_name** (*str*): The name of the summary.
- **summary_filename** (*str*): Base filename of the summary.
- **check_for_warnings** (*bool*): If true include warnings as well as errors.

The purpose of this op is to produce a summary of the HED validation errors in a file.

`__init__`(*parameters*)

Constructor for the summarize hed validation operation.

**Parameters**
**parameters** (*dict*) – Dictionary with the parameter values for required and optional parameters.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

### Methods

| | |
|---|---|
| *__init__*(parameters) | Constructor for the summarize hed validation operation. |
| *check_parameters*(parameters) | Verify that the parameters meet the operation specification. |
| *do_op*(dispatcher, df, name[, sidecar]) | Validate the dataframe with the accompanying sidecar, if any. |

### Attributes

| |
|---|
| PARAMS |
| SUMMARY_TYPE |

**check_parameters**(*parameters*)

Verify that the parameters meet the operation specification.

**Parameters**

**parameters** (*dict*) – Dictionary of parameters for this operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

**do_op**(*dispatcher*, *df*, *name*, *sidecar=None*)

Validate the dataframe with the accompanying sidecar, if any.

**Parameters**

- **dispatcher** (Dispatcher) – Manages the operation I/O.
- **df** (*DataFrame*) – The DataFrame to be validated.
- **name** (*str*) – Unique identifier for the dataframe – often the original file path.
- **sidecar** (Sidecar *or file-like*) – Usually needed unless only HED tags in HED column of event file.

**Returns**

A copy of df

**Return type**

DataFrame

**Side effect:**

Updates the relevant summary.

## hed.tools.remodeling.operations.summarize_sidecar_from_events_op

Create a JSON sidecar from column values in a collection of tabular files.

### Classes

| | |
|---|---|
| *EventsToSidecarSummary*(sum_op) | |
| *SummarizeSidecarFromEventsOp*(parameters) | Create a JSON sidecar from column values in a collection of tabular files. |

## hed.tools.remodeling.operations.summarize_sidecar_from_events_op.EventsToSidecarSummary

**class EventsToSidecarSummary**(*sum_op*)

Bases: *BaseSummary*

**__init__**(*sum_op*)

### Methods

| | |
|---|---|
| *__init__*(sum_op) | |
| dump_summary(filename, summary) | |
| *get_details_dict*(summary_info) | Return the summary-specific information. |
| get_individual(summary_details[, separately]) | |
| *get_summary*([individual_summaries]) | Return a summary dictionary with the information. |
| *get_summary_details*([include_individual]) | Return a dictionary with the details for individual files and the overall dataset. |
| get_text_summary([individual_summaries]) | |
| get_text_summary_details([include_individual]) | |
| *merge_all_info*() | Merge summary information from all the files. |
| save(save_dir[, file_formats, ...]) | |
| *update_summary*(new_info) | Update the summary for a given tabular input file. |

### Attributes

| |
|---|
| DISPLAY_INDENT |
| INDIVIDUAL_SUMMARIES_PATH |

**get_details_dict**(*summary_info*)

Return the summary-specific information.

**Parameters**

**summary_info** (TabularSummary) – Summary to return info from

### Notes

Abstract method be implemented by each individual context summary.

**get_summary**(*individual_summaries='separate'*)

Return a summary dictionary with the information.

**Parameters**
**individual_summaries** (*str*) – "separate", "consolidated", or "none"

**Returns**
dict - dictionary with "Dataset" and "Individual files" keys.

**Notes: The individual_summaries value is processed as follows**

- "separate" individual summaries are to be in separate files
- "consolidated" means that the individual summaries are in same file as overall summary
- "none" means that only the overall summary is produced.

**get_summary_details**(*include_individual=True*)

Return a dictionary with the details for individual files and the overall dataset.

**Parameters**
**include_individual** (*bool*) – If True, summaries for individual files are included.

**Returns**
dict - a dictionary with 'Dataset' and 'Individual files' keys.

### Notes

- The 'Dataset' value is either a string or a dictionary with the overall summary.
- **The 'Individual files' value is dictionary whose keys are file names and values are** their corresponding summaries.

Users are expected to provide merge_all_info and get_details_dict to support this.

**merge_all_info**()

Merge summary information from all the files.

**Returns**
Consolidated summary of information.

**Return type**
*TabularSummary*

**update_summary**(*new_info*)

Update the summary for a given tabular input file.

**Parameters**
**new_info** (*dict*) – A dictionary with the parameters needed to update a summary.

### Notes

- The summary needs a "name" str and a "df".

## hed.tools.remodeling.operations.summarize_sidecar_from_events_op.SummarizeSidecarFromEventsOp

**class SummarizeSidecarFromEventsOp**(*parameters*)

Bases: *BaseOp*

Create a JSON sidecar from column values in a collection of tabular files.

**Required remodeling parameters:**

- **summary_name** (*str*): The name of the summary.
- **summary_filename** (*str*): Base filename of the summary.
- **skip_columns** (*list*): Names of columns to skip in the summary.
- **value_columns** (*list*): Names of columns to treat as value columns rather than categorical columns.

The purpose is to produce a JSON sidecar template for annotating a dataset with HED tags.

**__init__**(*parameters*)

Constructor for summarize sidecar from events operation.

**Parameters**

**parameters** (*dict*) – Dictionary with the parameter values for required and optional parameters.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

### Methods

| | |
|---|---|
| *__init__*(parameters) | Constructor for summarize sidecar from events operation. |
| *check_parameters*(parameters) | Verify that the parameters meet the operation specification. |
| *do_op*(dispatcher, df, name[, sidecar]) | Extract a sidecar from events file. |

### Attributes

| |
|---|
| PARAMS |
| SUMMARY_TYPE |

**check_parameters**(*parameters*)

Verify that the parameters meet the operation specification.

**Parameters**

**parameters** (*dict*) – Dictionary of parameters for this operation.

**Raises**

- **KeyError** –
  - If a required parameter is missing.
  - If an unexpected parameter is provided.
- **TypeError** –
  - If a parameter has the wrong type.

**do_op**(*dispatcher*, *df*, *name*, *sidecar=None*)

Extract a sidecar from events file.

**Parameters**

- **dispatcher** (Dispatcher) – The dispatcher object for managing the operations.
- **df** (*DataFrame*) – The tabular file to be remodeled.
- **name** (*str*) – Unique identifier for the dataframe – often the original file path.
- **sidecar** (Sidecar *or file-like*) – Not needed for this operation.

**Returns**

A copy of df.

**Return type**

DataFrame

**Side effect:**

Updates the associated summary if applicable.

## hed.tools.remodeling.operations.valid_operations

The valid operations for the remodeling tools.

### 3.4.4 hed.tools.util

Data and file handling utilities.

#### Modules

| | |
|---|---|
| hed.tools.util.data_util | Data handling utilities involving dataframes. |
| hed.tools.util.hed_logger | Logger class with messages organized by key |
| hed.tools.util.io_util | Utilities for generating and handling file names. |
| hed.tools.util.schema_util | |

#### 3.4.4.1 hed.tools.util.data_util

Data handling utilities involving dataframes.

#### Functions

| | |
|---|---|
| add_columns(df, column_list[, value]) | Add specified columns to df if not there. |
| check_match(ds1, ds2[, numeric]) | Check two Pandas data series have the same values. |
| delete_columns(df, column_list) | Delete the specified columns from a dataframe. |
| delete_rows_by_column(df, value[, column_list]) | Delete rows where columns have this value. |
| get_eligible_values(values, values_included) | Return a list of the items from values that are in values_included or None if no values_included |
| get_indices(df, column, start, stop) | |
| get_key_hash(key_tuple) | Calculate a hash key for tuple of values. |
| get_new_dataframe(data) | Get a new dataframe representing a tsv file. |
| get_row_hash(row, key_list) | Get a hash key from key column values for row. |
| get_value_dict(tsv_path[, key_col, value_col]) | Get a dictionary of two columns of a dataframe. |
| make_info_dataframe(col_info, selected_col) | Get a dataframe from selected columns. |
| reorder_columns(data, col_order[, skip_missing]) | Create a new dataframe with columns reordered. |
| replace_values(df[, values, replace_value, ...]) | Replace string values in specified columns. |
| separate_values(values, target_values) | Get target values from the target_values list. |
| tuple_to_range(tuple_list, inclusion) | |

#### hed.tools.util.data_util.add_columns

**add_columns**(*df*, *column_list*, *value='n/a'*)

Add specified columns to df if not there.

**Parameters**

- **df** (*DataFrame*) – Pandas dataframe.
- **column_list** (*list*) – List of columns to append to the dataframe.
- **value** (*str*) – Default fill value for the column.

### hed.tools.util.data_util.check_match

**check_match**(*ds1*, *ds2*, *numeric=False*)

Check two Pandas data series have the same values.

**Parameters**

- **ds1** (*DataSeries*) – Pandas data series to check.
- **ds2** (*DataSeries*) – Pandas data series to check.
- **numeric** (*bool*) – If true, treat as numeric and do close-to comparison.

**Returns**

Error messages indicating the mismatch or empty if the series match.

**Return type**

list

### hed.tools.util.data_util.delete_columns

**delete_columns**(*df*, *column_list*)

Delete the specified columns from a dataframe.

**Parameters**

- **df** (*DataFrame*) – Pandas dataframe from which to delete columns.
- **column_list** (*list*) – List of candidate column names for deletion.

#### Notes

- The deletion of columns is done in place.
- This does not raise an error if df does not have a column in the list.

### hed.tools.util.data_util.delete_rows_by_column

**delete_rows_by_column**(*df*, *value*, *column_list=None*)

Delete rows where columns have this value.

**Parameters**

- **df** (*DataFrame*) – Pandas dataframe from which to delete rows.
- **value** (*str*) – Specified value to indicate row should be deleted.
- **column_list** (*list*) – List of columns to search for value.

#### Notes

- All values are converted to string before testing.
- Deletion is done in place.

### hed.tools.util.data_util.get_eligible_values

**get_eligible_values**(*values*, *values_included*)

Return a list of the items from values that are in values_included or None if no values_included

**Parameters**

- **values** (*list*) – List of strings against which to test.
- **values_included** (*list*) – List of items to be selected from values if they are present.

**Returns**
list of selected values or None if values_included is empty or None.

**Return type**
list

### hed.tools.util.data_util.get_indices

**get_indices**(*df*, *column*, *start*, *stop*)

### hed.tools.util.data_util.get_key_hash

**get_key_hash**(*key_tuple*)

Calculate a hash key for tuple of values.

**Parameters**
**key_tuple** (*tuple, list*) – The key values in the correct order for lookup.

**Returns**
A hash key for the tuple.

**Return type**
int

### hed.tools.util.data_util.get_new_dataframe

**get_new_dataframe**(*data*)

Get a new dataframe representing a tsv file.

**Parameters**
**data** (*DataFrame or str*) – DataFrame or filename representing a tsv file.

**Returns**

**A dataframe containing the contents of the tsv file or if data was**
a DataFrame to start with, a new copy of the DataFrame.

**Return type**
DataFrame

**Raises**
*HedFileError* –

- A filename is given, and it cannot be read into a Dataframe.

### hed.tools.util.data_util.get_row_hash

**get_row_hash**(*row*, *key_list*)

Get a hash key from key column values for row.

**Parameters**

- **row** (*DataSeries*) –
- **key_list** (*list*) –

**Returns**
Hash key constructed from the entries of row in the columns specified by key_list.

**Return type**
str

**Raises**
*HedFileError* –

- If row doesn't have all the columns in key_list HedFileError is raised.

### hed.tools.util.data_util.get_value_dict

**get_value_dict**(*tsv_path*, *key_col='file_basename'*, *value_col='sampling_rate'*)

Get a dictionary of two columns of a dataframe.

**Parameters**

- **tsv_path** (*str*) – Path to a tsv file with a header row to be read into a DataFrame.
- **key_col** (*str*) – Name of the column which should be the key.
- **value_col** (*str*) – Name of the column which should be the value.

**Returns**
Dictionary with key_col values as the keys and the corresponding value_col values as the values.

**Return type**
dict

**Raises**
*HedFileError* –

- When tsv_path does not correspond to a file that can be read into a DataFrame.

### hed.tools.util.data_util.make_info_dataframe

**make_info_dataframe**(*col_info*, *selected_col*)

Get a dataframe from selected columns.

**Parameters**

- **col_info** (*dict*) – Dictionary of dictionaries of column values and counts.
- **selected_col** (*str*) – Name of the column used as top level key for col_info.

**Returns**

**A two-column dataframe with first column containing values from the**
dictionary whose key is selected_col and whose second column are the corresponding counts. The returned value is None if selected_col is not a top-level key in col_info.

**Return type**
dataframe

### hed.tools.util.data_util.reorder_columns

**reorder_columns**(*data*, *col_order*, *skip_missing=True*)

Create a new dataframe with columns reordered.

**Parameters**

- **data** (*DataFrame, str*) – Dataframe or filename of dataframe whose columns are to be reordered.
- **col_order** (*list*) – List of column names in desired order.
- **skip_missing** (*bool*) – If true, col_order columns missing from data are skipped, otherwise error.

**Returns**
A new reordered dataframe.

**Return type**
DataFrame

**Raises**
*HedFileError* –

- If col_order contains columns not in data and skip_missing is False.
- If data corresponds to a filename from which a dataframe cannot be created.

### hed.tools.util.data_util.replace_values

**replace_values**(*df*, *values=None*, *replace_value='n/a'*, *column_list=None*)

Replace string values in specified columns.

**Parameters**

- **df** (*DataFrame*) – Dataframe whose values will be replaced.
- **values** (*list, None*) – List of strings to replace. If None, only empty strings are replaced.
- **replace_value** (*str*) – String replacement value.

- **column_list** (*list, None*) – List of columns in which to do replacement. If None all columns are processed.

**Returns**
number of values replaced.

**Return type**
int

### hed.tools.util.data_util.separate_values

**separate_values**(*values*, *target_values*)

Get target values from the target_values list.

**Parameters**

- **values** (*list*) – List of values to be tested.
- **target_values** – List of desired values.

### hed.tools.util.data_util.tuple_to_range

**tuple_to_range**(*tuple_list*, *inclusion*)

#### 3.4.4.2 hed.tools.util.hed_logger

Logger class with messages organized by key

### Classes

| | |
|---|---|
| *HedLogger*([name]) | Log status messages organized by key. |

### hed.tools.util.hed_logger.HedLogger

**class HedLogger**(*name=None*)

Bases: object

Log status messages organized by key.

**__init__**(*name=None*)

Constructor for the HED logger.

**Parameters**
**name** (*str*) – Identifying name of the logger.

### Methods

| | |
|---|---|
| *__init__*([name]) | Constructor for the HED logger. |
| add(key, msg[, level, also_print]) | |
| get_log(key) | |
| get_log_keys() | |
| *get_log_string*([level]) | Return the log as a string, with entries separated by newlines. |

**get_log_string**(*level=None*)

Return the log as a string, with entries separated by newlines.

**Parameters**
**level** (*str or None*) – Include only the entries from this level. If None, do all.

**Returns**
The log as a string separated by newlines.

**Return type**
str

#### 3.4.4.3 hed.tools.util.io_util

Utilities for generating and handling file names.

### Functions

| | |
|---|---|
| *check_filename*(test_file[, name_prefix, ...]) | Return True if correct extension, suffix, and prefix. |
| *clean_filename*(filename) | Replaces invalid characters with under-bars |
| *extract_suffix_path*(path, prefix_path) | Return the suffix of path after prefix path has been removed. |
| *get_allowed*(value[, allowed_values, starts_with]) | Return the portion of the value that matches a value in allowed_values or None if no match. |
| *get_dir_dictionary*(dir_path[, name_prefix, ...]) | Create dictionary directory paths keys. |
| *get_file_list*(root_path[, name_prefix, ...]) | Return paths satisfying various conditions. |
| *get_filtered_by_element*(file_list, elements) | Filter a file list by whether the base names have a substring matching any of the members of elements. |
| *get_filtered_list*(file_list[, name_prefix, ...]) | Get list of filenames satisfying the criteria. |
| *get_path_components*(root_path, this_path) | Get a list of the remaining components after root path. |
| *get_timestamp*() | |
| *make_path*(root_path, sub_path, filename) | Get path for a file, verifying all components exist. |
| *parse_bids_filename*(file_path) | Split a filename into BIDS-relevant components. |

### hed.tools.util.io_util.check_filename

**check_filename**(*test_file*, *name_prefix=None*, *name_suffix=None*, *extensions=None*)

Return True if correct extension, suffix, and prefix.

**Parameters**

- **test_file** (*str*) – Path of filename to test.
- **name_prefix** (*list, str, None*) – An optional name_prefix or list of prefixes to accept for the base filename.
- **name_suffix** (*list, str, None*) – An optional name_suffix or list of suffixes to accept for the base file name.
- **extensions** (*list, str, None*) – An optional extension or list of extensions to accept for the extensions.

**Returns**
True if file has the appropriate format.

**Return type**
bool

#### Notes

- Everything is converted to lower case prior to testing so this test should be case-insensitive.
- None indicates that all are accepted.

### hed.tools.util.io_util.clean_filename

**clean_filename**(*filename*)

Replaces invalid characters with under-bars

**Parameters**
**filename** (*str*) – source filename

**Returns**
The filename with anything but alphanumeric, period, hyphens, and under-bars removed.

**Return type**
str

### hed.tools.util.io_util.extract_suffix_path

**extract_suffix_path**(*path*, *prefix_path*)

Return the suffix of path after prefix path has been removed.

**Parameters**

- **path** (*str*) –
- **prefix_path** (*str*) –

**Returns**
Suffix path.

**Return type**
str

### Notes

- This function is useful for creating files within BIDS datasets

## hed.tools.util.io_util.get_allowed

**get_allowed**(*value*, *allowed_values=None*, *starts_with=True*)

Return the portion of the value that matches a value in allowed_values or None if no match.

**Parameters**

- **value** (*str*) – value to be matched.
- **allowed_values** (*list, str, or None*) – Values to match.
- **starts_with** (*bool*) – If true match is done at beginning of string, otherwise the end.

### Notes

- match is done in lower case.

## hed.tools.util.io_util.get_dir_dictionary

**get_dir_dictionary**(*dir_path*, *name_prefix=None*, *name_suffix=None*, *extensions=None*, *skip_empty=True*, *exclude_dirs=None*)

Create dictionary directory paths keys.

**Parameters**

- **dir_path** (*str*) – Full path of the directory tree to be traversed (no ending slash).
- **name_prefix** (*str, None*) – An optional name_prefix for the base filename.
- **name_suffix** (*str, None*) – An optional name_suffix for the base file name.
- **extensions** (*list, None*) – An optional list of file extensions.
- **skip_empty** (*bool*) – Do not put entry for directories that have no files.
- **exclude_dirs** (*list*) – List of directories to skip

**Returns**
Dictionary with directories as keys and file lists values.

**Return type**
dict

### hed.tools.util.io_util.get_file_list

**get_file_list**(*root_path*, *name_prefix=None*, *name_suffix=None*, *extensions=None*, *exclude_dirs=None*)

Return paths satisfying various conditions.

**Parameters**

- **root_path** (*str*) – Full path of the directory tree to be traversed (no ending slash).
- **name_prefix** (*str, None*) – An optional name_prefix for the base filename.
- **name_suffix** (*str, None*) – The name_suffix of the paths to be extracted.
- **extensions** (*list, None*) – A list of extensions to be selected.
- **exclude_dirs** (*list, None*) – A list of paths to be excluded.

**Returns**
The full paths.

**Return type**
list

### hed.tools.util.io_util.get_filtered_by_element

**get_filtered_by_element**(*file_list*, *elements*)

Filter a file list by whether the base names have a substring matching any of the members of elements.

**Parameters**

- **file_list** (*list*) – List of file paths to be filtered.
- **elements** (*list*) – List of strings to use as filename filters.

**Returns**
The list only containing file paths whose filenames match a filter.

**Return type**
list

### hed.tools.util.io_util.get_filtered_list

**get_filtered_list**(*file_list*, *name_prefix=None*, *name_suffix=None*, *extensions=None*)

Get list of filenames satisfying the criteria.

Everything is converted to lower case prior to testing so this test should be case-insensitive.

**Parameters**

- **file_list** (*list*) – List of files to test.
- **name_prefix** (*str*) – Optional name_prefix for the base filename.
- **name_suffix** (*str*) – Optional name_suffix for the base filename.
- **extensions** – Optional list of file extensions (allows two periods (.tsv.gz))

### hed.tools.util.io_util.get_path_components

get_path_components(*root_path*, *this_path*)

Get a list of the remaining components after root path.

**Parameters**

- **root_path** (*str*) – A path (no trailing separator)
- **this_path** (*str*) – The path of a file or directory descendant of root_path

**Returns**
A list with the remaining elements directory components to the file.

**Return type**
list or None

Notes: this_path must be a descendant of root_path.

### hed.tools.util.io_util.get_timestamp

get_timestamp()

### hed.tools.util.io_util.make_path

make_path(*root_path*, *sub_path*, *filename*)

Get path for a file, verifying all components exist.

**Parameters**

- **root_path** (*str*) – path of the root directory.
- **sub_path** (*str*) – sub-path relative to the root directory.
- **filename** (*str*) – filename of the file.

**Returns**
A valid realpath for the specified file.

**Return type**
str

Notes: This function is useful for creating files within BIDS datasets

### hed.tools.util.io_util.parse_bids_filename

parse_bids_filename(*file_path*)

Split a filename into BIDS-relevant components.

**Parameters**
**file_path** (*str*) – Path to be parsed.

**Returns**
BIDS suffix name. str: File extension (including the .). dict: Dictionary with key-value pair being (entity type, entity value).

**Return type**
str

**Raises**

*HedFileError* –

- If filename does not conform to name-value_suffix format.

### Notes

- splits into BIDS suffix, extension, and a dictionary of entity name-value pairs.

#### 3.4.4.4 hed.tools.util.schema_util

### Functions

| | |
|---|---|
| *flatten_schema*(hed_schema[, skip_non_tag]) | turns a schema into a 3 column dataframe. |

### hed.tools.util.schema_util.flatten_schema

**flatten_schema**(*hed_schema*, *skip_non_tag=False*)

turns a schema into a 3 column dataframe. :param hed_schema: the schema to flatten :type hed_schema: HedSchema :param skip_non_tag: Skips all sections except tag :type skip_non_tag: bool

## 3.4.5 hed.tools.visualization

### Modules

| |
|---|
| *hed.tools.visualization.tag_word_cloud* |
| *hed.tools.visualization.word_cloud_util* |

#### 3.4.5.1 hed.tools.visualization.tag_word_cloud

### Functions

| | |
|---|---|
| *create_wordcloud*(word_dict[, mask_path, ...]) | Takes a word dict and returns a generated word cloud object |
| *load_and_resize_mask*(mask_path[, width, height]) | Load a mask image and resize it according to given dimensions. |
| *summary_to_dict*(summary[, transform, adjustment]) | Converts a HedTagSummary json dict into the word cloud input format |

### hed.tools.visualization.tag_word_cloud.create_wordcloud

create_wordcloud(*word_dict*, *mask_path=None*, *background_color=None*, *width=400*, *height=200*, ***kwargs*)

Takes a word dict and returns a generated word cloud object

**Parameters**

- **word_dict** (*dict*) – words and their frequencies
- **mask_path** (*str or None*) – The path of the mask file
- **background_color** (*str or None*) – If None, transparent background.
- **width** (*int*) – width in pixels
- **height** (*int*) – height in pixels
- **kwargs** (*kwargs*) – Any other parameters WordCloud accepts, overrides default values where relevant.

**Returns**

**The generated cloud.**
Use .to_file to save it out as an image.

**Return type**
word_cloud(WordCloud)

**Raises**
**ValueError** – An empty dictionary was passed

### hed.tools.visualization.tag_word_cloud.load_and_resize_mask

load_and_resize_mask(*mask_path*, *width=None*, *height=None*)

Load a mask image and resize it according to given dimensions.

The image is resized maintaining aspect ratio if only width or height is provided.

Returns None if no mask_path.

**Parameters**

- **mask_path** (*str*) – The path to the mask image file.
- **width** (*int, optional*) – The desired width of the resized image. If only width is provided, the image is scaled to maintain its original aspect ratio. Defaults to None.
- **height** (*int, optional*) – The desired height of the resized image. If only height is provided, the image is scaled to maintain its original aspect ratio. Defaults to None.

**Returns**
The loaded and processed mask image as a numpy array with binary values (0 or 255).

**Return type**
numpy.ndarray

#### hed.tools.visualization.tag_word_cloud.summary_to_dict

**summary_to_dict**(*summary*, *transform=<ufunc 'log10'>*, *adjustment=5*)

Converts a HedTagSummary json dict into the word cloud input format

**Parameters**

- **summary** (*dict*) – The summary from a summarize hed tags op
- **transform** (*func*) – The function to transform the number of found tags Default log10
- **adjustment** (*int*) – Value added after transform.

**Returns**
a dict of the words and their occurrence count

**Return type**
word_dict(dict)

**Raises**
**KeyError** – A malformed dictionary was passed

### 3.4.5.2 hed.tools.visualization.word_cloud_util

#### Functions

| | |
|---|---|
| *default_color_func*(word, font_size, ...[, ...]) | |
| *random_color_darker*([word, font_size, ...]) | Random color generation func |

#### hed.tools.visualization.word_cloud_util.default_color_func

**default_color_func**(*word*, *font_size*, *position*, *orientation*, *random_state=None*, ***kwargs*)

#### hed.tools.visualization.word_cloud_util.random_color_darker

**random_color_darker**(*word=None*, *font_size=None*, *position=None*, *orientation=None*, *font_path=None*, *random_state=None*)

Random color generation func

#### Classes

| | |
|---|---|
| *ColormapColorFunc*([colormap, color_range, ...]) | |

#### hed.tools.visualization.word_cloud_util.ColormapColorFunc

class **ColormapColorFunc**(*colormap='nipy_spectral'*, *color_range=(0.0, 0.5)*, *color_step_range=(0.15, 0.25)*)

Bases: `object`

**__init__**(*colormap='nipy_spectral'*, *color_range=(0.0, 0.5)*, *color_step_range=(0.15, 0.25)*)

Initialize a word cloud color generator.

**Parameters**

- **colormap** (*str, optional*) – The name of the matplotlib colormap to use for generating colors. Defaults to 'nipy_spectral'.
- **color_range** (*tuple of float, optional*) – A tuple containing the minimum and maximum values to use from the colormap. Defaults to (0.0, 0.5).
- **color_step_range** (*tuple of float, optional*) – A tuple containing the minimum and maximum values to step through the colormap. Defaults to (0.15, 0.25). This is the speed at which it goes through the range chosen. .25 means it will go through 1/4 of the range each pick.

#### Methods

| | |
|---|---|
| *__init__*([colormap, color_range, ...]) | Initialize a word cloud color generator. |
| color_func(word, font_size, position, ...[, ...]) | |

## 3.5 hed.validator

Validation of HED tags.

#### Modules

| | |
|---|---|
| *hed.validator.def_validator* | |
| *hed.validator.hed_validator* | This module contains the HedValidator class which is used to validate the tags in a HED string or a file. |
| *hed.validator.onset_validator* | |
| *hed.validator.sidecar_validator* | |
| *hed.validator.spreadsheet_validator* | |
| *hed.validator.tag_validator* | This module is used to validate the HED tags as strings. |
| *hed.validator.tag_validator_util* | Utilities to support HED validation. |

### 3.5.1 hed.validator.def_validator

#### Classes

| | |
|---|---|
| *DefValidator*([def_dicts, hed_schema]) | Handles validating Def/ and Def-expand/, as well as Temporal groups: Onset, Inset, and Offset |

#### 3.5.1.1 hed.validator.def_validator.DefValidator

**class DefValidator**(*def_dicts=None*, *hed_schema=None*)

Bases: *DefinitionDict*

Handles validating Def/ and Def-expand/, as well as Temporal groups: Onset, Inset, and Offset

**__init__**(*def_dicts=None*, *hed_schema=None*)

Initialize for definitions in hed strings.

**Parameters**

- **def_dicts** (*list or DefinitionDict or str*) – DefinitionDicts containing the definitions to pass to baseclass
- **hed_schema** (*HedSchema or None*) – Required if passing strings or lists of strings, unused otherwise.

#### Methods

| | |
|---|---|
| *__init__*([def_dicts, hed_schema]) | Initialize for definitions in hed strings. |
| *add_definitions*(def_dicts[, hed_schema]) | Add definitions from dict(s) to this dict. |
| *check_for_definitions*(hed_string_obj[, ...]) | Check string for definition tags, adding them to self. |
| *construct_def_tag*(hed_tag) | Identify def/def-expand tag contents in the given HedTag. |
| *construct_def_tags*(hed_string_obj) | Identify def/def-expand tag contents in the given string. |
| *get*(def_name) | Get the definition entry for the definition name. |
| *get_as_strings*(def_dict) | Convert the entries to strings of the contents |
| *items*() | Returns the dictionary of definitions |
| *validate_def_tags*(hed_string_obj[, ...]) | Validate Def/Def-Expand tags. |
| *validate_onset_offset*(hed_string_obj) | Validate onset/offset |

#### Attributes

| | |
|---|---|
| *issues* | Returns issues about duplicate definitions. |

**add_definitions**(*def_dicts*, *hed_schema=None*)

Add definitions from dict(s) to this dict.

**Parameters**

- **def_dicts** (*list, DefinitionDict, or dict*) –

**DefinitionDict or list of DefinitionDicts/strings/dicts whose**
definitions should be added.

Note dict form expects DefinitionEntries in the same form as a DefinitionDict

- **hed_schema** (HedSchema *or None*) – Required if passing strings or lists of strings, unused otherwise.

**Raises**
**TypeError** –

- Bad type passed as def_dicts

**check_for_definitions**(*hed_string_obj*, *error_handler=None*)

Check string for definition tags, adding them to self.

**Parameters**

- **hed_string_obj** (HedString) – A single hed string to gather definitions from.
- **error_handler** (ErrorHandler *or None*) – Error context used to identify where definitions are found.

**Returns**
List of issues encountered in checking for definitions. Each issue is a dictionary.

**Return type**
list

**construct_def_tag**(*hed_tag*)

Identify def/def-expand tag contents in the given HedTag.

**Parameters**
**hed_tag** (HedTag) – The hed tag to identify definition contents in

**construct_def_tags**(*hed_string_obj*)

Identify def/def-expand tag contents in the given string.

**Parameters**
**hed_string_obj** (HedString) – The hed string to identify definition contents in

**get**(*def_name*)

Get the definition entry for the definition name.

Not case-sensitive

**Parameters**
**def_name** (*str*) – Name of the definition to retrieve.

**Returns**
Definition entry for the requested definition.

**Return type**
*DefinitionEntry*

**static get_as_strings**(*def_dict*)

Convert the entries to strings of the contents

**Parameters**
**def_dict** (DefinitionDict *or dict*) – A dict of definitions

**Returns**
str): definition name and contents

**Return type**
dict(str

**property issues**

Returns issues about duplicate definitions.

**items**()

Returns the dictionary of definitions

Alias for .defs.items()

**Returns**
DefinitionEntry}): A list of definitions

**Return type**
def_entries({str

**validate_def_tags**(*hed_string_obj*, *tag_validator=None*)

Validate Def/Def-Expand tags.

**Parameters**

- **hed_string_obj** (HedString) – The hed string to process.
- **tag_validator** (TagValidator) – Used to validate the placeholder replacement.

**Returns**
Issues found related to validating defs. Each issue is a dictionary.

**Return type**
list

**validate_onset_offset**(*hed_string_obj*)

Validate onset/offset

**Parameters**
**hed_string_obj** (HedString) – The hed string to check.

**Returns**
A list of issues found in validating onsets (i.e., out of order onsets, unknown def names).

**Return type**
list

### 3.5.2 hed.validator.hed_validator

This module contains the HedValidator class which is used to validate the tags in a HED string or a file. The file types include .tsv, .txt, and .xlsx. To get the validation issues after creating a HedValidator class call the get_validation_issues() function.

### Classes

| | |
|---|---|
| *HedValidator*(hed_schema[, def_dicts, ...]) | Top level validation of HED strings. |

#### 3.5.2.1 hed.validator.hed_validator.HedValidator

**class HedValidator**(*hed_schema*, *def_dicts=None*, *definitions_allowed=False*)

Bases: `object`

Top level validation of HED strings.

**__init__**(*hed_schema*, *def_dicts=None*, *definitions_allowed=False*)

Constructor for the HedValidator class.

**Parameters**

- **hed_schema** (*HedSchema or HedSchemaGroup*) – HedSchema object to use for validation.
- **def_dicts** (*DefinitionDict or list or dict*) – the def dicts to use for validation
- **definitions_allowed** (*bool*) – If False, flag definitions found as errors

### Methods

| | |
|---|---|
| *__init__*(hed_schema[, def_dicts, ...]) | Constructor for the HedValidator class. |
| run_basic_checks(hed_string, allow_placeholders) | |
| run_full_string_checks(hed_string) | |
| *validate*(hed_string, allow_placeholders[, ...]) | Validate the string using the schema |

**validate**(*hed_string*, *allow_placeholders*, *error_handler=None*)

Validate the string using the schema

**Parameters**

- **hed_string** (*HedString*) – the string to validate
- **allow_placeholders** (*bool*) – allow placeholders in the string
- **error_handler** (*ErrorHandler or None*) – the error handler to use, creates a default one if none passed

**Returns**

A list of issues for hed string

**Return type**

issues (list of dict)

### 3.5.3 hed.validator.onset_validator

#### Classes

| | |
|---|---|
| *OnsetValidator*() | Validates onset/offset pairs. |

#### 3.5.3.1 hed.validator.onset_validator.OnsetValidator

**class OnsetValidator**

Bases: object

Validates onset/offset pairs.

**__init__**()

##### Methods

| | |
|---|---|
| *__init__*() | |
| *validate_temporal_relations*(hed_string_obj) | Validate onset/offset/inset tag relations |

**validate_temporal_relations**(*hed_string_obj*)

Validate onset/offset/inset tag relations

**Parameters**
**hed_string_obj** (HedString) – The hed string to check.

**Returns**
A list of issues found in validating onsets (i.e., out of order onsets, unknown def names).

**Return type**
list

### 3.5.4 hed.validator.sidecar_validator

#### Classes

| | |
|---|---|
| *SidecarValidator*(hed_schema) | |

#### 3.5.4.1 hed.validator.sidecar_validator.SidecarValidator

**class SidecarValidator**(*hed_schema*)

Bases: object

**__init__**(*hed_schema*)

Constructor for the HedValidator class.

**Parameters**
**hed_schema** (HedSchema) – HED schema object to use for validation.

### Methods

| | |
|---|---|
| *__init__*(hed_schema) | Constructor for the HedValidator class. |
| *validate*(sidecar[, extra_def_dicts, name, ...]) | Validate the input data using the schema |
| *validate_structure*(sidecar, error_handler) | Validate the raw structure of this sidecar. |

### Attributes

| |
|---|
| reserved_category_values |
| reserved_column_names |

**validate**(*sidecar*, *extra_def_dicts=None*, *name=None*, *error_handler=None*)

Validate the input data using the schema

**Parameters**

- **sidecar** (Sidecar) – Input data to be validated.
- **extra_def_dicts** (*list or* DefinitionDict) – extra def dicts in addition to sidecar
- **name** (*str*) – The name to report this sidecar as
- **error_handler** (ErrorHandler) – Error context to use. Creates a new one if None

**Returns**
A list of issues associated with each level in the HED string.

**Return type**
issues (list of dict)

**validate_structure**(*sidecar*, *error_handler*)

Validate the raw structure of this sidecar.

**Parameters**

- **sidecar** (Sidecar) – the sidecar to validate
- **error_handler** (ErrorHandler) – The error handler to use for error context

**Returns**
A list of issues found with the structure

**Return type**
issues(list)

### 3.5.5 hed.validator.spreadsheet_validator

#### Classes

| | |
|---|---|
| *SpreadsheetValidator*(hed_schema) | |

#### 3.5.5.1 hed.validator.spreadsheet_validator.SpreadsheetValidator

**class SpreadsheetValidator**(*hed_schema*)

Bases: object

**__init__**(*hed_schema*)

Constructor for the HedValidator class.

**Parameters**
**hed_schema** (HedSchema) – HED schema object to use for validation.

#### Methods

| | |
|---|---|
| *__init__*(hed_schema) | Constructor for the HedValidator class. |
| *validate*(data[, def_dicts, name, error_handler]) | Validate the input data using the schema |

**validate**(*data*, *def_dicts=None*, *name=None*, *error_handler=None*)

Validate the input data using the schema

**Parameters**

- **data** (*BaseInput or pd.DataFrame*) – Input data to be validated. If a dataframe, it is assumed to be assembled already.
- **def_dicts** (*list of DefDict or DefDict*) – all definitions to use for validation
- **name** (*str*) – The name to report errors from this file as
- **error_handler** (ErrorHandler) – Error context to use. Creates a new one if None

**Returns**
A list of issues for hed string

**Return type**
issues (list of dict)

### 3.5.6 hed.validator.tag_validator

This module is used to validate the HED tags as strings.

### Classes

| | |
|---|---|
| *TagValidator*(hed_schema) | Validation for individual HED tags. |

#### 3.5.6.1 hed.validator.tag_validator.TagValidator

**class TagValidator**(*hed_schema*)

Bases: object

Validation for individual HED tags.

**__init__**(*hed_schema*)

Constructor for the Tag_Validator class.

**Parameters**
**hed_schema** (HedSchema) – A HedSchema object.

**Returns**
A Tag_Validator object.

**Return type**
*TagValidator*

### Methods

| | |
|---|---|
| *__init__*(hed_schema) | Constructor for the Tag_Validator class. |
| *check_capitalization*(original_tag) | Report warning if incorrect tag capitalization. |
| *check_count_tag_group_parentheses*(hed_string) | Report unmatched parentheses. |
| *check_delimiter_issues_in_hed_string*(hed_string) | Report missing commas or commas in value tags. |
| *check_for_invalid_extension_chars*(original_tag) | Report invalid characters in extension/value. |
| *check_for_placeholder*(original_tag[, ...]) | Report invalid placeholder characters. |
| *check_for_required_tags*(tags) | Report missing required tags. |
| *check_invalid_character_issues*(hed_string, ...) | Report invalid characters. |
| *check_multiple_unique_tags_exist*(tags) | Report if multiple identical unique tags exist |
| *check_tag_exists_in_schema*(original_tag) | Report invalid tag or doesn't take a value. |
| *check_tag_formatting*(original_tag) | Report repeated or erroneous slashes. |
| *check_tag_invalid_chars*(original_tag, ...) | Report invalid characters in the given tag. |
| *check_tag_level_issue*(original_tag_list, ...) | Report tags incorrectly positioned in hierarchy. |
| *check_tag_requires_child*(original_tag) | Report if tag is a leaf with 'requiredTag' attribute. |
| *check_tag_unit_class_units_are_valid*(...[, ...]) | Report incorrect unit class or units. |
| *check_tag_value_class_valid*(original_tag[, ...]) | Report an invalid value portion. |
| *run_all_tags_validators*(tags) | Validate the multi-tag properties in a hed string. |
| *run_hed_string_validators*(hed_string_obj[, ...]) | Basic high level checks of the hed string |
| *run_individual_tag_validators*(original_tag) | Runs the hed_ops on the individual tags. |
| *run_tag_level_validators*(original_tag_list, ...) | Run hed_ops at each level in a HED string. |
| *validate_value_class_type*(...) | Report invalid unit or valid class values. |

### Attributes

| |
|---|
| CAMEL_CASE_EXPRESSION |
| CLOSING_GROUP_CHARACTER |
| COMMA |
| DEFAULT_ALLOWED_PLACEHOLDER_CHARS |
| INVALID_STRING_CHARS |
| INVALID_STRING_CHARS_PLACEHOLDERS |
| OPENING_GROUP_CHARACTER |
| TAG_ALLOWED_CHARS |
| pattern_doubleslash |

**check_capitalization**(*original_tag*)

Report warning if incorrect tag capitalization.

**Parameters**
**original_tag** (HedTag) – The original tag used to report the warning.

**Returns**
Validation issues. Each issue is a dictionary.

**Return type**
list

**check_count_tag_group_parentheses**(*hed_string*)

Report unmatched parentheses.

**Parameters**
**hed_string** (*str*) – A hed string.

**Returns**
A list of validation list. Each issue is a dictionary.

**Return type**
list

**check_delimiter_issues_in_hed_string**(*hed_string*)

Report missing commas or commas in value tags.

**Parameters**
**hed_string** (*str*) – A hed string.

**Returns**
A validation issues list. Each issue is a dictionary.

**Return type**
list

**check_for_invalid_extension_chars**(*original_tag*)

Report invalid characters in extension/value.

**Parameters**
**original_tag** (HedTag) – The original tag that is used to report the error.

**Returns**
Validation issues. Each issue is a dictionary.

**Return type**
list

**check_for_placeholder**(*original_tag*, *is_definition=False*)

Report invalid placeholder characters.

**Parameters**

- **original_tag** (HedTag) – The HedTag to be checked
- **is_definition** (*bool*) – If True, placeholders are allowed.

**Returns**
Validation issues. Each issue is a dictionary.

**Return type**
list

### Notes

- Invalid placeholder may appear in the extension/value portion of a tag.

**check_for_required_tags**(*tags*)

Report missing required tags.

**Parameters**
**tags** (*list*) – HedTags containing the tags.

**Returns**
Validation issues. Each issue is a dictionary.

**Return type**
list

**check_invalid_character_issues**(*hed_string*, *allow_placeholders*)

Report invalid characters.

**Parameters**

- **hed_string** (*str*) – A hed string.
- **allow_placeholders** – Allow placeholder and curly brace characters

**Returns**
Validation issues. Each issue is a dictionary.

**Return type**
list

### Notes

- **Invalid tag characters are defined by TagValidator.INVALID_STRING_CHARS or** TagValidator.INVALID_STRING_CHARS_PLACEHOLDERS

**check_multiple_unique_tags_exist**(*tags*)

Report if multiple identical unique tags exist

A unique Term can only appear once in a given HedString. Unique terms are terms with the 'unique' property in the schema.

**Parameters**
**tags** (*list*) – HedTags containing the tags.

**Returns**
Validation issues. Each issue is a dictionary.

**Return type**
list

**check_tag_exists_in_schema**(*original_tag*)

Report invalid tag or doesn't take a value.

**Parameters**
**original_tag** (HedTag) – The original tag that is used to report the error.

**Returns**
Validation issues. Each issue is a dictionary.

**Return type**
list

**check_tag_formatting**(*original_tag*)

Report repeated or erroneous slashes.

**Parameters**
**original_tag** (HedTag) – The original tag that is used to report the error.

**Returns**
Validation issues. Each issue is a dictionary.

**Return type**
list

**check_tag_invalid_chars**(*original_tag*, *allow_placeholders*)

Report invalid characters in the given tag.

**Parameters**

- **original_tag** (HedTag) – The original tag that is used to report the error.
- **allow_placeholders** (*bool*) – Allow placeholder characters(#) if True.

**Returns**
Validation issues. Each issue is a dictionary.

**Return type**
list

**check_tag_level_issue**(*original_tag_list*, *is_top_level*, *is_group*)

Report tags incorrectly positioned in hierarchy.

**Parameters**

- **original_tag_list** (*list*) – HedTags containing the original tags.
- **is_top_level** (*bool*) – If True, this group is a "top level tag group"
- **is_group** (*bool*) – If true group should be contained by parenthesis

**Returns**
Validation issues. Each issue is a dictionary.

**Return type**
list

### Notes

- Top-level groups can contain definitions, Onset, etc tags.

**check_tag_requires_child**(*original_tag*)

Report if tag is a leaf with 'requiredTag' attribute.

**Parameters**
**original_tag** (HedTag) – The original tag that is used to report the error.

**Returns**
Validation issues. Each issue is a dictionary.

**Return type**
list

**check_tag_unit_class_units_are_valid**(*original_tag*, *report_as=None*, *error_code=None*)

Report incorrect unit class or units.

**Parameters**

- **original_tag** (HedTag) – The original tag that is used to report the error.
- **report_as** (HedTag) – Report errors as coming from this tag, rather than original_tag.
- **error_code** (*str*) – Override error codes to this

**Returns**
Validation issues. Each issue is a dictionary.

**Return type**
list

**check_tag_value_class_valid**(*original_tag*, *report_as=None*, *error_code=None*)

Report an invalid value portion.

**Parameters**

- **original_tag** (HedTag) – The original tag that is used to report the error.
- **report_as** (HedTag) – Report errors as coming from this tag, rather than original_tag.
- **error_code** (*str*) – Override error codes to this

**Returns**
Validation issues.

**Return type**
list

**run_all_tags_validators**(*tags*)

Validate the multi-tag properties in a hed string.

**Parameters**
**tags** (*list*) – A list containing the HedTags in a HED string.

**Returns**
The validation issues associated with the tags in a HED string. Each issue is a dictionary.

**Return type**
list

### Notes

- Multi-tag properties include required tags.

**run_hed_string_validators**(*hed_string_obj*, *allow_placeholders=False*)

Basic high level checks of the hed string

**Parameters**

- **hed_string_obj** (HedString) – A HED string.
- **allow_placeholders** – Allow placeholder and curly brace characters

**Returns**
The validation issues associated with a HED string. Each issue is a dictionary.

**Return type**
list

### Notes

- Used for basic invalid characters or bad delimiters.

**run_individual_tag_validators**(*original_tag*, *allow_placeholders=False*, *is_definition=False*)

Runs the hed_ops on the individual tags.

**Parameters**

- **original_tag** (HedTag) – A original tag.
- **allow_placeholders** (*bool*) – Allow value class or extensions to be placeholders rather than a specific value.
- **is_definition** (*bool*) – This tag is part of a Definition, not a normal line.

**Returns**
The validation issues associated with the top-level tags. Each issue is dictionary.

**Return type**
list

**run_tag_level_validators**(*original_tag_list*, *is_top_level*, *is_group*)

Run hed_ops at each level in a HED string.

**Parameters**

- **original_tag_list** (*list*) – A list containing the original HedTags.
- **is_top_level** (*bool*) – If True, this group is a "top level tag group".
- **is_group** (*bool*) – If true, group is contained by parenthesis.

**Returns**
The validation issues associated with each level in a HED string.

**Return type**
list

#### Notes

- This is for the top-level, all groups, and nested groups.
- This can contain definitions, Onset, etc tags.

**validate_value_class_type**(*unit_or_value_portion*, *valid_types*)

Report invalid unit or valid class values.

**Parameters**

- **unit_or_value_portion** (*str*) – The value portion to validate.
- **valid_types** (*list*) – The names of value class or unit class types (e.g. dateTime or dateTimeClass).

**Returns**
True if this is one of the valid_types validators.

**Return type**
type_valid (bool)

### 3.5.7 hed.validator.tag_validator_util

Utilities to support HED validation.

#### Functions

| | |
|---|---|
| *is_clock_face_time*(time_string) | Check if a valid HH:MM time string. |
| *is_date_time*(date_time_string) | Check if the specified string is a valid datetime. |
| *validate_numeric_value_class*(numeric_string) | Checks to see if valid numeric value. |
| *validate_text_value_class*(text_string) | Placeholder for eventual text value class validation |

#### 3.5.7.1 hed.validator.tag_validator_util.is_clock_face_time

**is_clock_face_time**(*time_string*)

Check if a valid HH:MM time string.

**Parameters**
**time_string** (*str*) – A time string.

**Returns**
True if the time string is valid. False, if otherwise.

**Return type**
bool

#### Notes

- This is deprecated and has no expected use going forward.

#### 3.5.7.2 hed.validator.tag_validator_util.is_date_time

`is_date_time`(*date_time_string*)

Check if the specified string is a valid datetime.

**Parameters**
**date_time_string** (*str*) – A datetime string.

**Returns**
True if the datetime string is valid. False, if otherwise.

**Return type**
bool

#### Notes

- ISO 8601 datetime string.

#### 3.5.7.3 hed.validator.tag_validator_util.validate_numeric_value_class

`validate_numeric_value_class`(*numeric_string*)

Checks to see if valid numeric value.

**Parameters**
**numeric_string** (*str*) – A string that should be only a number with no units.

**Returns**
True if the numeric string is valid. False, if otherwise.

**Return type**
bool

#### 3.5.7.4 hed.validator.tag_validator_util.validate_text_value_class

`validate_text_value_class`(*text_string*)

Placeholder for eventual text value class validation

**Parameters**
**text_string** (*str*) – Text class.

**Returns**
True

**Return type**
bool

CHAPTER

FOUR

# INDICES AND TABLES

- genindex
- modindex
- search

# PYTHON MODULE INDEX

## h

# INDEX

## Symbols

## A



## B



## C

## D

## E



## F

## G

## H

## I



## K



## L



## M

## N



## O

## P



## Q



## R

## S

## T

## U



## V

## W