**HEDTools Validation Utilities: Validation Tools**

**User Guide**

Table of Contents

[1. Getting Started with HED Validation Tools 1](#_Toc443060193)

[1.1 Overview 1](#_Toc443060194)

[1.2 Requirements 1](#_Toc443060195)

[1.3 Installation 1](#_Toc443060196)

[2. Using the HED Validation Utilities GUI 1](#_Toc443060197)

[2.1. Validating tab-delimited tags 2](#_Toc443060198)

[2.2. Remapping tab-delimited tags 4](#_Toc443060199)

[2.3. Updating tagging vocabulary 5](#_Toc443060200)

# 1. Getting Started with HED Validation Tools

## 1.1 Overview

*HEDTools Validation Utilities* is a MATLAB Toolbox designed to help users validate events or other data elements using a predefined, but extensible, hierarchically structured annotation language. Although *HED Validation Utilities* can be used in very general contexts, this document describes the MATLAB interface used specifically for validating events in EEG data. In the case of EEG, users validate the events that occur during an EEG experiment using the HED 2 hierarchical event description language as the vocabulary.

## 1.2 Requirements

*HED Validation Utilities* has been tested with MATLAB 2015b on 64-bit Windows 8. No other toolboxes are required.

## 1.3 Installation

You can run *HED Validation Utilities* as a standalone toolbox. It can be downloaded by clicking the *Download ZIP* button at <https://github.com/VisLab/HEDTools>. The code is in the validation/matlab directory.

# 2. Using the HED Validation Utilities GUI

You can launch *HED Validation Utilities* GUIfrom the *pop\_hedtagvalidate.m* file or a series of top-level validation functions (*remapTSVTags*, *validateTSVTags*). This guide focuses on using *HED Validation Utilities* through MATLAB. *HED Validation Utilities* assumes the availability of a hierarchically structured vocabulary for validating tags.

## 2.1. Validating tab-delimited tags

The *HED Validation Utilities validateTSVTags* function allows you to validate tags in a tab-delimited file against a tagging vocabulary.

The following example illustrates the simplest use of *validatTSVTags* for a tab-delimited file. The tab-delimited file can have a header and any number of columns and rows.

**Example 2.1:** Validate the tags in the tab-delimited file.

[errors, warnings, extensions, remap] = validateTSVTags('LSIE\_01\_Indoor\_all\_events.tsv, 3, 'hasHeader', false);

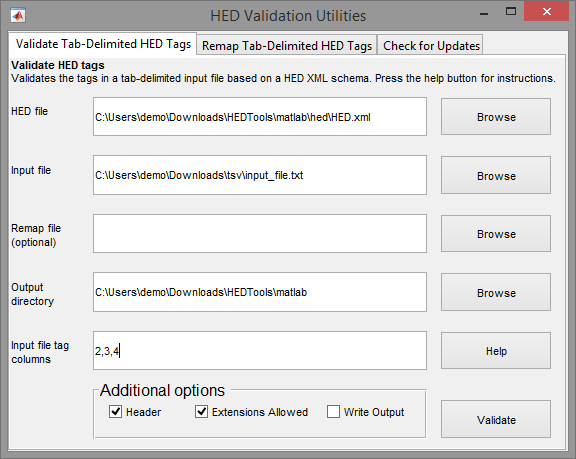
The *errors* return parameter is a cell array containing all of the validation errors. The *warnings* return parameter is a cell array containing all of the validation warnings. The *extensions* return parameter is a cell array containing all of the extension allowed validation warnings. Each cell in these three return parameters are associated with a particular line that raised an issue. The *remap* return parameter is a cell array containing all of the unique tags that generated validation errors. Please refer to section 2.2 for explanation of how the remapping works.

The *validateTSVTags* function validates the tags in column *3* from the tab-delimited file LSIE\_01\_Indoor\_all\_events.tsv.

**Table 2.1:** A summary of arguments for *validateTSVTags*.

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| tsvFile | Required | The name or the path of a tab-delimited file containing HED tags associated with a particular study. |
| tsvTagColumns | Required | The columns that contain the study HED tags. The columns can be a scalar value or a vector (e.g. 2 or [2,3,4]). |
| 'extensionAllowed' | Name-Value | True (default) if descendants of extension allowed tags are accepted which will generate extension warnings. False if they are not accepted which will generate errors. |
| 'hasHeader' | Name-Value | True (default) if the tab-delimited file containing the study HED tags has a header. The header will be skipped and not validated. False if the file doesn't have a header. |
| 'hedXML' | Name-Value | The name or the path of the XML file containing all of the HED tags and their attributes (tagging vocabulary). |
| 'outputDirectory' | Name-Value | A directory where the validation output is written to if the *'writeOuput'* argument is true. There will be four separate files generated, one containing the validation errors, one containing the validation warnings, one containing the extension allowed validation warnings, and a remap file. The default directory will be the directory that contains the tab-delimited file *'tsvFile'*. |
| 'writeOutput' | Name-Value | True if the validation output is written to the workspace and a set of files in the same directory, false (default) if the validation output is only written to the workspace. |

**Figure 2.1:** The tag validation menu for *validateTSVTags*.



The top section of the menu allows you to browse for the tagging vocabulary (HED.xml), the tab-delimited input file, an optional remap file, the directory where the validation output is written to a set of files, and the tab-delimited input file tag columns. The bottom box designates additional validation options.

The *Header* argument controls whether or not the tab-delimited input file has a header. The *Extension Allowed* argument controls whether or not the validation accepts tags that are descendants of extension allowed tags. If accepted (checked) then each applicable tag will generate an extension allowed warning, if not accepted (unchecked) then each tag will generate an error. The *Write Output* argument will determine if the validation output is written to a set of files in addition to the workspace. If unchecked the validation output is only written to the workspace. Press the *Validate* button when the parameters have been set.

## 2.2. Remapping tab-delimited tags

The *HED Validation Utilities remapTSVTags* function allows you to remap (replace) old tags with new tags in a tab-delimited file. This is useful when the tags in the tab-delimited file are from an older tagging vocabulary or there are simply typo errors. The idea here is that the remap file allows users to replace the tags in an automated way.

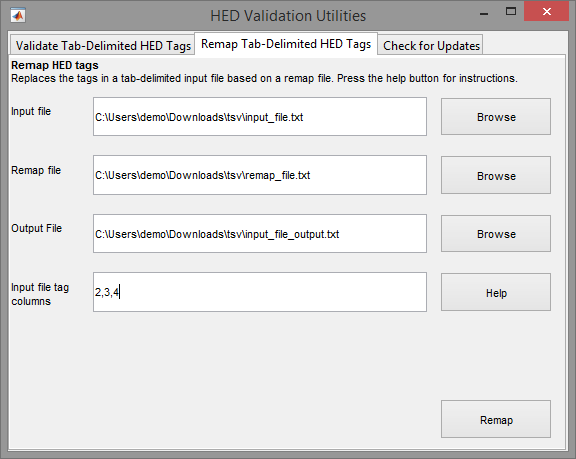
**Example 2.2:** Remaps the tags in the tab-delimited file.

remapTSVTags('BCIT\_Driving\_HED\_tag\_spec\_v25\_remap.tsv', 'BCIT\_Driving\_HED\_tag\_spec\_v25.tsv', [3,4,5,6]);

**Table 2.2:** A summary of arguments for *remapTSVTags*.

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| remapFile | Required | The name or the path of the remap file containing the mappings of old HED tags to new HED tags. This file is a two column tab-delimited file with the old HED tags in the first column and the new HED tags in the second column. |
| tsvFile | Required | The name or the path of a tab-delimited file containing HED tags associated with a particular study. |
| tsvTagColumns | Required | The columns that contain the study HED tags. The columns can be a scalar value or a vector (e.g. 2 or [2,3,4]). |
| 'outputFile' | Name-Value | The name or the path to the file that the output is written to. The output file will be a tab-delimited file consisting of the new HED tags which replaced the old HED tags in each specified columns. The default file name will be the original tab-delimited *tsvFile* filename with \_update appended to. |

**Figure 2.2:** The tag validation menu for *remapTSVTags*.

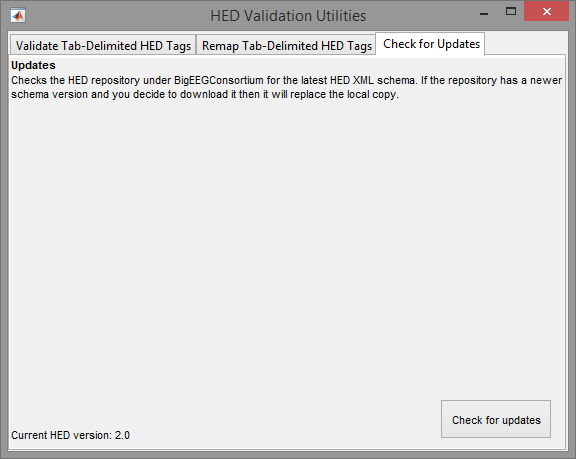


The menu allows you to browse for the tab-delimited input file, a remap file, the directory where the remap output file is written to, and the tab-delimited input file tag columns. Press the *Remap* button when the parameters have been set.

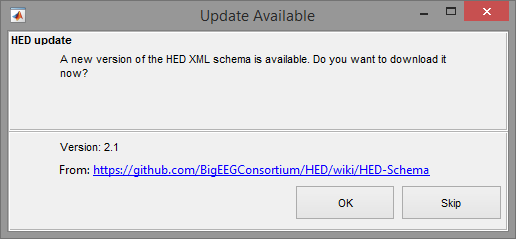
## 2.3. Updating tagging vocabulary

You can check to see if there is a newer tagging vocabulary available. This allows you to avoid downloading the HED *Validation Tools* every time the tagging vocabulary has been updated.

**Figure 2.3:** Checking for updates.



The menu contains the current version of the tagging vocabulary at the bottom left. Press the *Check for updates* button to check if there is a newer version available. If a newer version is available then the menu below will appear.

**Figure 2.4:** Update available.

The menu tells you the latest version that is available for download. You have the option to download it or skip downloading it. Clicking on the hyperlink above the two buttons allows you to see what the latest version looks like before downloading it. If you decide to download it click the *OK* button.