
WDL-AID

DavyCats

Dec 06, 2019

TABLE OF CONTENTS

1	Preparing your WDL files	1
1.1	Excluding inputs	1
1.2	Metadata	2
2	Running WDL-AID	3
2.1	Writing to a file	3
2.2	Fallback/default values	3
2.3	Custom description and category keys	3
2.4	Keeping original categories for required inputs	4
2.5	Custom templates	4
2.6	Extra data	4
2.7	Strict mode	4
3	Custom templates	5
3.1	Minimalistic Example	5
4	Changelog	7
4.1	v0.1.1	7
4.2	v0.1.0	7
5	Welcome to WDL-AID's documentation!	9
5.1	Quick start	9
5.1.1	Installation	9
5.1.2	Basic usage	9
5.2	Reporting bugs and feature requests	10
5.3	Index	10
	Index	11

PREPARING YOUR WDL FILES

Before the documentation can be generated, each input in your [WDL](#) file must be given a description and category. This is done using WDL's [parameter_meta](#) section:

```
parameter_meta {  
  name_of_input: {  
    description: "Some description of the input",  
    category: "some category"  
  }  
}
```

These fields (`description` and `category` may also be called differently, but you will have to set some additional options when running WDL-AID, see [Custom description and category keys](#).

WDL-AID will separate the inputs by category, so each category may be rendered in its own section. Required inputs are automatically detected and assigned the `required` category, overwriting the one noted in the `parameter_meta` section.

The default template supports the following categories:

- `required`
- `common`
- `advanced`
- `other`

1.1 Excluding inputs

In some cases there may be inputs which should not be included in the documentation, eg. when using a sub-workflow which provides options which make no sense in the context of the overarching workflow. You can tell WDL_AID to omit certain inputs by adding the following to your workflow's [meta](#) section:

```
WDL_AID: {  
  exclude: ["input_name", "call.input_name"]  
}
```

The inputs added here may be of the workflow or task containing the meta section or from any call made inside of the workflow. Be sure to use the input_names qualified relative to this workflow, ie. `input_name` for inputs of the task/workflow itself, `call_name.input_name` for inputs of calls, `call_name.sub_call_name.input_name` for calls inside of sub-workflows, etc.

1.2 Metadata

The `meta` section of your workflow may also be used to pass additional information along to WDL-AID. The entire `meta` section of the workflow that WDL-AID gets called on is passed to the template unaltered. WDL-AID will also retrieve all authors from the `meta` sections of every called task and workflow and pass these along.

The default template supports the following `meta` section entries:

- `description` (only for the root workflow)
- `authors` (also for sub-workflows and tasks)

This is expected to be an object containing the following fields for each author:

- `name`
- `email` (optional)
- `organization` (optional)

eg.

```
meta {
  authors: [
    {
      name: "Eddard Stark",
      email: "StarkNed@winterfell.westeros",
      organization: "The North"
    }, {
      name: "Jon Snow",
      email: "j.snow@nightswatch.westeros",
      organization: "The Night's Watch"
    }
  ]
}
```

RUNNING WDL-AID

WDL-AID can be run with the following command:

```
wdl-aid <workflow.wdl>
```

This will print the generated documentation to `stdout`. This will be markdown formatted text when using the default template.

2.1 Writing to a file

To write the output to a file the following option can be used:

-o OUTPUT, --output OUTPUT
The file to write the generated documentation to.

2.2 Fallback/default values

If no description or category is defined then WDL-AID will fallback to a default value. By default these values equal `???` and `other` respectively. You may override these fallback values using the following options:

--fallback-description FALLBACK_DESCRIPTION
The fallback value for when no description is defined for a given input.

--fallback-category FALLBACK_CATEGORY
The fallback value for when no category is defined for a given input.

In some cases a `parameter_meta` entry may be defined, but it does not contain the an object with a description item. By default the fallback values will get used in these cases. However, alternatively you can use the entirety of the defined `parameter_meta` entry as description value using the following flag:

--fallback-description-to-object
Use the entire `parameter_meta` object as description if the description key is not found.

2.3 Custom description and category keys

In case your `parameter_meta` entries use different keys than `description` and `category` to provide the description and category (respectively) of the inputs, you can use the following options to inform WDL-AID of which keys to look for:

- c** CATEGORY_KEY, **--category-key** CATEGORY_KEY
The key used in the `parameter_meta` sections for the input category.
- d** DESCRIPTION_KEY, **--description-key** DESCRIPTION_KEY
The key used in the `parameter_meta` section for the input description.

2.4 Keeping original categories for required inputs

If you wish to retain the categories noted in the `parameter_meta` sections for the required input, rather than having the overwritten with `required` then you can use the following flag:

2.5 Custom templates

You can provide a custom template using the following option. This template should be a [Jinja2](#) template.

- t** TEMPLATE, **--template** TEMPLATE
A Jinja2 template to use for rendering the documentation.

See [Custom templates](#) for more details on making a custom template.

2.6 Extra data

It is possible to pass extra data along to the template. This can be done by providing the following option with a json file which contains this extra data.

- e** EXTRA, **--extra** EXTRA
A JSON file with additional data to be passed to the `jinja2` rendering engine.

2.7 Strict mode

WDL-AID has an option to run in a “strict” mode. This entails that WDL-AID will error if any inputs are missing a `parameter_meta` section. This may be useful as part of CI testing, allowing you to ensure that all inputs will always be documented.

- strict**
Error if the `parameter_meta` entry is missing for any inputs.

CUSTOM TEMPLATES

WDL-AID uses [Jinja2](#) to generate documentation files. By default a template is provided to Jinja2 which comes packaged with WDL-AID. This default template will result in a markdown file. If you wish to generate a differently formatted file (eg. html) or simply want to change the content of the document then you can provide a custom Jinja2 template. This custom template can be passed to WDL-AID using the `-t` option.

The following variables are made available to the template:

- `workflow_name`: The name of the workflow.
- `workflow_file`: The path given as input to WDL-AID.
- `workflow_authors`: A list of author information, taken from the `authors` field in the meta section. If this field does not contain a list its value will be wrapped in one.
- `workflow_all_authors`: A list of author information taken from the `authors` fields from the workflow and called sub-workflows and tasks.
- `workflow_meta`: A direct copy of the workflow's meta section.
- `excluded_inputs`: A list of fully-qualified inputs which will be available, but will be excluded from the rendering process.
- `wdl_aid_version`: The version of WDL-AID used
- Per category a list of dictionaries. Each of these dictionaries will describe an input and contains the following keys:
 - `name`: The (fully qualified) name of the input.
 - `type`: The WDL value type of the input (eg. `String?` or `Pair[Int, Boolean]`).
 - `default`: The default value of the input. If an input has no default, then `None`.
 - `description`: The description of the input as specified in the `parameter_meta` sections in the WDL file(s).
- `extra`: Whatever value is contained within the JSON file provided though the `-e` option, otherwise `None`.

3.1 Minimalistic Example

The following is a small example of a template that could be used with WDL-AID.

```
<html>
<head>
  <title>{{ workflow_name }}</title>
  <style>
```

(continues on next page)

(continued from previous page)

```
    ul { list-style: none; }
    li { background: #e5e5e5; padding: 10px; }
    li:nth-child(odd) { background: #f0f0f0; }
    dt { font-weight: bold }
  </style>
</head>
<body>
  <h1>{{ workflow_name }}</h1>
  {{ workflow_description }}
  <h2>Required Inputs</h2>

  <ul>
    {% for ri in required|sort(attribute='name') %}
      <li>
        <dl>
          <dt>name</dt>
          <dd>{{ ri.name }}</dd>
          <dt>type</dt>
          <dd>{{ ri.type }}</dd>
          <dt>default value</dt>
          <dd>{{ ri.default }}</dd>
          <dt>description</dt>
          <dd>{{ ri.description }}</dd>
        </dl>
      </li>
    {% endfor %}
  </ul>
</body>
</html>
```

CHANGELOG

4.1 v0.1.1

- Inputs without a default will now be given a `None` value in the default field passed to `jinja2`, instead of a string containing `None`. This should not impact generated documents (unless specific logic dealing with `None` values is used), as `jinja` will still render `None` values as `None`.

4.2 v0.1.0

- initial release

WELCOME TO WDL-AID'S DOCUMENTATION!

Generate documentation for the inputs of [WDL](#) workflows, based on the [parameter_meta](#) information defined in the WDL file.

WDL-AID is developed by the Sequencing Analysis Support Core at the [Leiden University Medical Center](#).

5.1 Quick start

5.1.1 Installation

WDL-AID can be installed using:

```
pip install wdl-aid
```

5.1.2 Basic usage

Running WDL-AID requires the following steps:

1. Add [parameter_meta](#) sections to your tasks and workflows. These should be objects containing both a description and category:

```
parameter_meta {  
  input_name: {  
    description: "A description of what value should be provided and is what_  
→it is used for.",  
    category: "required"  
  }  
}
```

The available categories in the default template are:

- required
- common
- advanced
- other

Required inputs are automatically detected and their noted category will be overwritten with `required`.

2. Once installed, WDL-AID can be run using the following command:

```
wdl-aid <workflow.wdl> -o docs.md
```

This will generate the file `docs.md`, containing the generated documentation.

5.2 Reporting bugs and feature requests

Please report any bugs, issues or pull requests on the [github issue tracker](#).

5.3 Index

`genindex`

Symbols

- category-key CATEGORY_KEY
 - wdl-aid command line option, 3
- description-key DESCRIPTION_KEY
 - wdl-aid command line option, 4
- extra EXTRA
 - wdl-aid command line option, 4
- fallback-category FALLBACK_CATEGORY
 - wdl-aid command line option, 3
- fallback-description
 - FALLBACK_DESCRIPTION
 - wdl-aid command line option, 3
- fallback-description-to-object
 - wdl-aid command line option, 3
- output OUTPUT
 - wdl-aid command line option, 3
- strict
 - wdl-aid command line option, 4
- template TEMPLATE
 - wdl-aid command line option, 4
- c CATEGORY_KEY
 - wdl-aid command line option, 3
- d DESCRIPTION_KEY
 - wdl-aid command line option, 4
- e EXTRA
 - wdl-aid command line option, 4
- o OUTPUT
 - wdl-aid command line option, 3
- t TEMPLATE
 - wdl-aid command line option, 4

- strict, 4
- template TEMPLATE, 4
- c CATEGORY_KEY, 3
- d DESCRIPTION_KEY, 4
- e EXTRA, 4
- o OUTPUT, 3
- t TEMPLATE, 4

W

- wdl-aid command line option
 - category-key CATEGORY_KEY, 3
 - description-key DESCRIPTION_KEY, 4
 - extra EXTRA, 4
 - fallback-category
 - FALLBACK_CATEGORY, 3
 - fallback-description
 - FALLBACK_DESCRIPTION, 3
 - fallback-description-to-object, 3
 - output OUTPUT, 3