How to Document DIRAC and Extensions

Documentation is great, there should be more of it

André Sailer

CERN-EP-LCD

DIRAC User Workshop 2019
May 14–19, 2019

Disclaimer and Target Audience

This is *not*:

- a how-to for *restructured text* or `sphinx`

This *is*

- This is a collection of `things-I-wish-I-had-known-when-I-started`
- A description of tools to simplify documentation of DIRAC components

It might still be interesting if you ever have to document some other python based project

Table of Contents

Getting Started

Make an installation of DIRAC and the extensions you need, make sure `sphinx` is installed (`pip install`)

```
mkdir docs
cd docs
sphinx-quickstart
```

And answer the questions that come up

- Use separate source and build directories
- Sphinx Extensions: enable `autodoc`, `intersphinx`, and others to your liking

This generates the files: `Makefile`, `source/conf.py`, `source/index.rst`
Add those to your repository

conf.py

The `conf.py` contains the configuration for sphinx, including the default options, read the sphinx documentation for further information. You should probably read through the file once, and see if any option sounds interesting to you.

- Can execute anything inside, file is `execfile`'d when documentation is build
- The "official" DIRAC documentation theme: `html_theme = 'nature'`
- You can `inherit` to modify themes, e.g., reduce the width of the page, as we do for `iLCDirac`

```
def setup(app):
  """Add files to install."""
  app.add_stylesheet('theme_override.css')
```

Nitpicking

- Turn on the nitpicky mode

```
nitpicky = True
nitpick_ignore = [
    ('type', 'pattern'),
    ]
```

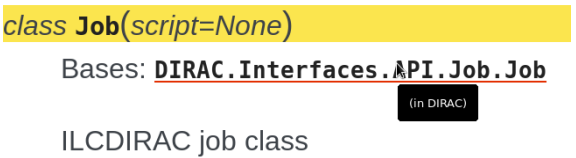
- This will produce warnings when a reference is not finding its target
- Warnings can be ignored, because they will show up, for example when defining parameter types.

Intersphinx

Using intersphinx, we can link to the sphinx documentation of other projects, like `DIRAC` and `Python`

```
intersphinx_mapping =
   {'DIRAC': ('https://dirac.readthedocs.io/en/rel-v6r21', None),
    'python': ('https://docs.python.org/2.7', None),
}
```

- For example, automatically links to DIRAC base class documentation



- Sphinx creates an `object.inv` file containing information about available targets, which is downloaded by intersphinx at build time

DiracDocTools

- `diracdoctools` are a collection of python commands to simplify the creation of the DIRAC documentation
  - API Reference
  - Command Reference
  - Config Templates
  - Release Notes
- located in `DIRAC/docs/diracdoctools`
- The commands are configured with a configuration file to avoid typing options on the command line that ≈never change
- Copy the file from `DIRAC/docs/docs.conf` and modify for your extension
- Follow the instructions how to incorporate into the `conf.py` here

  https://dirac.readthedocs.io/en/latest/DeveloperGuide/CodeDocumenting/BuildingDocumentation.html

API Reference

`dirac-docs-build-code.py`

- This script creates an `sphinx-autodoc` based tree of the respective source code
- For each python file it creates a file like this:

```
Client
======
.. automodule:: DIRAC.Core.Base.Client
   :members:
   :inherited-members:
   :undoc-members:
   :show-inheritance:
   :special-members: __init__
```

- During the creation of the documentation sphinx imports the module and creates documentation based on the docstrings
- Syntax errors, missing imports etc. will make the creation of the documentation fail

Extending Class Documentation

- For individual modules the autodoc directives can be overwritten or extended
- One Example in DIRAC/docs/diracdoctools/CustomizedDocs/
  - At build time, create documentation based on module constants

CustomDErrno.py

```
"""Additional docstring for the DErrno module."""
class CustomDErrno(object):  # pylint: disable=too-few-public-methods
  """Add the ERRNO constants to the docstring automatically."""
  def __init__(self):
    """Create a string containing restructured text documentation \
for all the defined error numbers.

    :param str module: full path to the module
    :param bool replace: whether to replace the full docstring or not
       if you replace it completely add the automodule commands etc.!
    """
    self.module = 'DIRAC.Core.Utilities.DErrno'
    self.replace = False
    self.doc_string = '\n'
    self.doc_string += 'ErrorCodes\n'
    self.doc_string += '----------\n'
    from DIRAC.Core.Utilities.DErrno import dErrorCode, dStrError
    for ec in sorted(set(dStrError) & set(dErrorCode)):
      # following the syntax for definition lists, bolding
      # the ERRNO so it looks like other module members
      self.doc_string += '\n%s.\\ **%s** = %d\n    %s' %
          (self.module, dErrorCode[ec], ec, dStrError[ec])

CUSTOMIZED_DOCSTRINGS['DIRAC.Core.Utilities.DErrno'] = CustomDErrno()# pylint: disable=undefined-variable
```

Command Reference

`dirac-docs-build-command.py`

- This script finds all the commands in your extension `*/scripts` folders, calls them with the `--help` flag set, and puts that into an `rst` file
- Based on pattern matching, commands can be sorted into separate indices, as done for the DIRAC UserGuide
- From the `docs.conf`

```
[commands.dms]
pattern = dms
title = Data Management
scripts =
ignore =
sectionPath = source/UserGuide/CommandReference/%(title)s
```

Concatenate ConfigTemplates

- `dirac-docs-concatenate-diraccfg.py`
- Finds all the ConfigTemplates and puts them in the *Systems* section of a CFG file

Agent/Service/Executor Options

- Don't Repeat Yourself (too much): define options and default values *once* in the `ConfigTemplate.cfg` of the system
  - At least ensure that the values in ConfigTemplate and source code are consistent
  - `DIRAC/ConfigurationSystem/test/Test_agentOptions.py` ensures consistency, allows exceptions
    - ⋆ Need same functionality for services
- Use `literalinclude` to include the options into the Agent/Service/Executor documentation
  - https://dirac.readthedocs.io/en/latest/CodeDocumentation/TransformationSystem/Agent/WorkflowTaskAgent.html

RPC'ed Client functions (in v7)

- Explicitly expose functions from a `ServiceHandler` in its `Client`:

```
from DIRAC.Core.Base.Client import Client, createClient
@createClient('WorkloadManagement/JobMonitoring')
class JobMonitoringClient(Client):
    ...
```

- Existing functions are not overwritten
- Adds functions to client classes on *import*
  - https://dirac.readthedocs.io/en/latest/CodeDocumentation/Core/Base/Client.html#DIRAC.Core.Base.Client.createClient

    getAtticJobParameters(**args*, ***kwargs*)

    Automatically created for the service function **export_getAtticJobParameters()**

    Parameters:
    - **jobID** – jobID
    - **parameters** – parameters
    - **rescheduleCycle** – rescheduleCycle

Release Notes

- If you have submitted a PR for DIRAC in the not too distant past you should have noticed the **BEGINRELEASENOTES/ENDRELEASENOTES** template
- Why do we bother (you) with that?
- `./diracdoctools/scripts/dirac-docs-get-release-notes.py`
- Picks up the content from the PR comments, and collates into the release notes structure
- It would also work for your extension, if it is hosted on GitHub or GitLab

Continuous Documentation (Testing)

- Add a test to your CI to ensure the documentation is building without warnings
- Fail PRs that introduce warnings or worse
- See https://github.com/DIRACGrid/DIRAC/blob/integration/.travis.d/checkDocs.sh

Some Existing Documentation

- https://dirac.readthedocs.io
- https://ilcdirac-doc.web.cern.ch
- https://lhcb-dirac.readthedocs.io
- https://github.com/DIRACGrid/VMDIRAC/pull/113

Open Issues

- Restructuring and improving documentation
- Add check for service options
- Document all options, also for Operations
- Turn on nitpicky for DIRAC, fix those warnings
- Add the `rst` file of the release notes to the DIRAC documentation
- ...

Conclusion and Outlook

- Creating the infrastructure for documenting DIRAC extensions is easy
- Improving the documentation is a great way of learning about (DIRAC) functionality
- I hope DIRAC v7 is released and adopted soon and to let automation take over the documentation checks