Rubin Observatory

Retro Theme Edition

Documentation Contribution Primer

Jonathan Sick
Slack/GitHub: @jonathansick

NSF AURA U.S. DEPARTMENT OF ENERGY Office of Science SLAC CHARLES AND LISA SIMONYI FUND FOR ARTS AND SCIENCES LSST CORPORATION

Hi, I'm Jonathan Sick

Rubin Observatory

- I'm a senior full-stack DevOps engineer on the SQuaRE team

Joined in 2015 after my PhD; this has been a journey!

I hope you'll also get a lot out of your career with Rubin

Building, operating, and supporting a variety of web apps, services, and infrastructure

- Rubin Science Platform
- SQuaSH/Sasquatch (observatory telemetry and pipelines metrics data engineering and monitoring)
- Documentation platform
- Rubin internal/developer experience tooling

What I'm most known/ responsible for

- How to get in touch
  - #dm-square on Slack for general conversation/questions about what we do
  - #dm-docs-support on Slack for documentation topics
  - Please avoid direct messages so we can share the support burden and share knowledge
- We love feedback and suggestions, but keep in mind it may take a while to get to if our current epic is focussed on something else

NSF
U.S. DEPARTMENT OF ENERGY | Office of Science

Docs
or it didn't happen

(a sentiment to strive for)

Rubin Observatory

U.S. DEPARTMENT OF ENERGY | Office of Science

What you will learn

- Gain a broad understanding of our documentation system and patterns.
- Learn how to write technical notes.
- Learn how to contribute to user guides and software documentation.

Outline

1. Documentation system
2. Creating documents and technical notes
3. ReStructuredText
4. Python API documentation (Numpydoc docstrings)
5. Writing user guides and Python software documentation
6. Learning how to write for users

Documentation system concepts: Docs-like-code

Rubin Observatory

- User guides for a software system are embedded in the software's GitHub repository (or in their own GitHub repository if the documentation covers multiple software repositories).
  - *Documentation is naturally versioned with the underlying software.*
  - *Documentation contribution review using Pull Requests.*
  - *Git and GitHub skills are necessary for technical writing.*
- Sphinx (sphinx-doc.org) is our main tool for building documentation sites.
  - *ReStructuredText is the primary format for writing content, along with Jupyter Notebooks for tutorials.*
  - *SQuaRE builds custom Sphinx extensions to improve doc production; inquire at #dm-docs-support on Slack.*
- Continuous integration (CI) and deployment (CD)
  - *Documentation sites are tested, built, and deployed to the web from the CI/CD server configured for a project (e.g. Jenkins or GitHub Actions).*
- Documentation is hosted on lsst.io, a platform powered by *LSST the Docs*.
  - *Documentation is publicly available on the web.*

Documentation system concepts: lsst.io

Rubin Observatory

- Documentation portal for search and browsing.
  - www.lsst.io
- Each documentation project is a subdomain.
  - E.g. pipelines.lsst.io, nb.lsst.io
- The "main" version of any documentation project is served from the root path.
  - E.g. pipelines.lsst.io/
  - The meaning of "main" is configurable, per-project
- Alternative versions are served with /v/{version}/ path prefixes.
  - E.g. pipelines.lsst.io/v/daily/, pipelines.lsst.io/v/v20_0_0/
- Dashboard for versions.
  - E.g. pipelines.lsst.io/v/

NSF
U.S. DEPARTMENT OF ENERGY | Office of Science

www.lsst.io

Documentation portal

Rubin Observatory

U.S. DEPARTMENT OF ENERGY | Office of Science

Key types of documentation

Rubin Observatory

Controlled documents

https://developer.lsst.io/project-docs/change-controlled-docs.html

- Requirements, specifications, design, test reports, and policies.
- Are approved by a control board.
- Officially published in DocuShare (docushare.lsstcorp.org)
- Many (but not all!) such documents are drafted on GitHub/lsst.io using LaTeX.

Series: LPM, LSE, LDM, etc.

Technical notes

https://developer.lsst.io/project-docs/technotes.html

- A medium for ground-up technical communication to your team, other teams, and beyond:
  Ideas, designs, proposals, experiments, literature reviews, and more. Even paper preprints.
- Combines the organization of a document archive with the immediacy and low overhead of a wiki.
- Migrate to guides (→) to document living systems and products.
- RST/Sphinx or LaTeX format

Technote series: DMTN, SQR, TSTN, SITCOMTN, SMTN, PSTN, RTN, etc.

Guides

- Multi-page websites that document a product, system, or service.
- Usually built with Sphinx (sphinx-doc.org) and Documenteer (documenteer.lsst.io).
- User-facing or developer/team-facing (or both!).
- Content types:
  - Conceptual guides
  - How-tos
  - Tutorials
  - References

NSF
U.S. DEPARTMENT OF ENERGY | Office of Science

Technical notes

Rubin Observatory

U.S. DEPARTMENT OF ENERGY | Office of Science

Let's make a technical note

Rubin Observatory

Step 1

Step 2

Step 3

U.S. DEPARTMENT OF ENERGY | Office of Science

Let's make a technical note

Rubin Observatory

U.S. DEPARTMENT OF ENERGY | Office of Science

Search or jump to...

lsst-sqre-testing / testn-10

Code | Issues | Pull r

master | 1 branch | 0

jonathansick and SQuaRE Bo

- .github/workflows
- _static
- lsstbib
- .gitignore
- COPYRIGHT
- LICENSE
- Makefile
- README.rst
- conf.py
- index.rst
- local.bib
- metadata.yaml
- requirements.txt

README.rst

testn-100 lsst.io | CI passing

Technical note demo

TESTN-100

Some of the possible applicatio
data processing or software de
a high-level overview complem
becoming the subject of a requ

Links:

- Publication URL: https://tes
- Alternative editions: https:/
- GitHub repository: https://
- Build system: https://githul

Search or jump to...

lsst-sqre-testing / testn-100

Code | Issues | Pull request

master | testn-100 / index.rst

jonathansick Initial commit ✓

1 contributor

62 lines (38 sloc) | 1.7 KB

tocdepth: 1

Note

This technote is not yet published.

Some of the possible applications for
development experiment. Announce a

Propose an architecture, possibly bec

© 2020 GitHub, Inc. Terms Privacy

Search or jump to...

lsst-sqre-testing / testn-100

Code | Issues | Pull requests

testn-100 / index.rst

Edit file | Preview changes

```
..
  Technote content.

  See https://developer.lsst.io/restr
  for a guide to reStructuredText wri

  Do not put the title, authors or ot
  those are automatically added.

  Use the following syntax for sectio

  Sections
  ========

  and

  Subsections
  -----------

  and

  Subsubsections
  ^^^^^^^^^^^^^^

  To add images, add the image file (
  _static/ directory. The reST syntax

  .. figure:: /_static/filename.ext
     :name: fig-label

     Caption text.

   Run: ``make html`` and ``open _bui
   See the README at https://github.c
   this repo's README for more info.
```

Commit changes

Update index.rst

Add an optional extended descript

jsick@lsst.org

Choose which email address to associate

Commit directly to the maste

Create a new branch for this

Large Synoptic Survey Telescope

TESTN-100: Technical note demonstration

Edition: master
Switch editions
Edit on GitHub

TESTN-

Jonathan Sick

Latest Revision

Note

This technote

Some of the p
data processi
a high-level o

Propose an arc

© Copyright 20
updated on Oc

Built with Sphi

Large Synoptic Survey Telescope

SQR-049: Science Platform token management design

Edition: master
Switch editions
Edit on GitHub

1 Abstract
2 Scope
3 Overview
4 Storage
5 API
6 Web UI
7 auth_request API
8 References

sqr-049.lsst.io

SQR-049: Science Platform token management design

Russ Allbery

Latest Revision: 2020-09-28

1 Abstract

Authentication tokens will be used by the science platform as web authentication credentials, for API and service calls from outside the Science Platform, and for internal service-to-service and notebook-to-service calls. This document lays out the technical design of the token management component, satisfying the requirements given in SQR-044.

2 Scope

This design covers the token component in isolation. The user management, group management, and quota components will be designed separately. That may result in some changes to this design as the rest of the system is built. If so, this document will be updated accordingly.

In addition to the requirements in SQR-044, see SQR-039 for a discussion of authentication and authorization for the Science Platform.

This specification will be implemented in Gafaelfawr, the authentication and authorization service for the Rubin Science Platform.

2.1 User metadata

Eventually, metadata about the user as opposed to their session or authentication token (full name, group memberships, UID, etc.) will be stored in a separate user management system that will provide an API to retrieve that information given an authentication token. However, that component of the overall identity management system has not yet been built. Instead, authorization is currently reliant on user metadata communicated via OAuth 2.0 or OpenID Connect and encoded in the resulting identity token.

Therefore, as an interim measure, user metadata is associated with each authentication token and stored with it. The token service provides an API to retrieve that metadata. The storage elements and APIs to support this are flagged below and should be considered temporary.

User-created tokens will, temporarily, inherit the user metadata (including group membership) from the session token used to create that user token. This means group membership will be encoded in the session data for that token and the token will have to be reissued to change that information, contrary to the design in SQR-044 and SQR-039. This will be fixed once the user metadata component is available as a separate service.

3 Overview

ReStructuredText

ReStructuredText is the format we use for writing most types of documentation:

- Python docstrings
- User guides (pipelines.lsst.io, developer.lsst.io, nb.lsst.io)
- Technical notes

Learn more

🔗 https://developer.lsst.io/restructuredtext/style.html

Rubin Observatory

U.S. DEPARTMENT OF ENERGY | Office of Science

🔗 https://developer.lsst.io/python/numpydoc.html

Rubin Observatory

Python docstrings

The docstrings in Python code are compiled into the API reference sections of software documentation guides, such as pipelines.lsst.io.

Following the docstring format is important for documentation builds.

NSF

U.S. DEPARTMENT OF ENERGY | Office of Science

Python docstrings

The basics

```
def upload():
    pass
```

Python docstrings

The basics

```
def upload():
    """Upload content to the server."""
    pass
```

Start and end the docstring with three double quotes. If docstring spans multiple lines put closing quotes on their own line.

Use imperative mood for summary sentence.

~~Uploads content to the server.~~

Wrap after the 79th character column.

Python docstrings

Function/method: Parameters

```
def upload(body):
    """Upload content to the server.

    Parameters
    ----------
    body : `bytes`
        The content to upload, already encoded as
        bytes.
    """
    pass
```

② Type description. Use single-backticks to link type names.

① Variable name

③ Indent description by four spaces.

For details: https://developer.lsst.io/python/numpydoc.html#parameters

Python docstrings

Function/method: Parameters

```
def upload(body, contentType='application/octet-stream'):
    """Upload content to the server.

    Parameters
    ----------
    body : `bytes`
        The content to upload, already encoded as bytes.
    contentType : `str`, optional
        The media type of the ``body``. Common types:

        ``application/octet-stream``
           A binary file (default).
        ``application/text``
           Text content.
        ``application/json``
           A JSON-formatted document.
    """
    pass
```

For details: https://developer.lsst.io/python/numpydoc.html#parameters

Python docstrings

Function/method: Returns

```
def upload(body, contentType='application/octet-stream'):
    """Upload content to the server.

    Parameters
    ----------
    [...]

    Returns
    -------
    url : `str`
        The canonical URL where the content can be accessed.
    """
    pass
```

Python docstrings

Function/method: Raises (exceptions)

```
def upload(body, contentType='application/octet-stream'):
    """Upload content to the server.

    Parameters
    ----------
    [...]

    Returns
    -------
    [...]

    Raises
    ------
    lsst.example.UploadError
        Raised if a connection cannot be made with the server, or if the
        server returns a 500 error.
    lsst.example.AuthError
        Raised if the client cannot be authenticated or is not
        authorized.
    """
    pass
```

Python docstrings

Class attribute docstrings

```
class Point:
    """A cartesian coordinate.

    Parameters
    ----------
    x : `float`
        The ``x``-axis coordinate.
    y : `float`
        The ``y``-axis coordinate.
    """

    x = None
    """The ``x``-axis coordinate (`float`)."""

    y = None
    """The ``y``-axis coordinate (`float`)."""

    def __init__(self, x, y):
        self.x = x
        self.y = y
```

Python docstrings

Class properties

```
class Point:
    """A cartesian coordinate.

    Parameters
    ----------
    x : `float`
        The ``x``-axis coordinate.
    y : `float`
        The ``y``-axis coordinate.
    """

    @property
    def x(self):
        """The ``x``-axis coordinate (`float`)."""
        return self._x


    def __init__(self, x, y):
        self._x = x
        self._y = y
```

🔗 https://developer.lsst.io/index.html#part-dm-stack

Rubin Observatory

Writing docs for Science Pipelines

pipelines.lsst.io

NSF

U.S. DEPARTMENT OF ENERGY | Office of Science

Science Pipelines documentation

Rubin Observatory

The Science Pipelines documentation project (pipelines.lsst.io) is exceptional because it is built from multiple source repositories:

github.com/lsst/pipelines_lsst_io

doc/ directories of individual packages

🔗 Further reading: https://developer.lsst.io/stack/documentation-system-overview.html

U.S. DEPARTMENT OF ENERGY | Office of Science

Building Science Pipelines documentation

Rubin Observatory

🔗 https://developer.lsst.io/stack/building-pipelines-lsst-io-locally.html

Mode 1: Building the whole documentation project.

developer.lsst.io

Clone and set up the pipelines_lsst_io repository
Install Documenteer, the documentation tooling
Building the pipelines_lsst_io site
Cleaning up built documentation
Adding a locally-developed package to the pipelines_lsst_io build
Further reading
Building pipelines.lsst.io with Jenkins (sqre/infra/documenteer)
Testing the LSST DM Stack with the Jenkins stack-os-matrix Job
Coverage Analysis
EUPS Tutorial
The LSST Software Build Tool
Adding a New Package to the Build
Managing license and copyright in Stack packages
Distributing Third-Party Packages with EUPS
Renaming a Stack package
GIT
Git Configuration Standards & Recommendations
Using Git LFS (Large File Storage) for data repositories
EDITORS
Emacs Configuration for LSST Development
SublimeText Configuration for LSST Development
Vim Configuration for LSST Development
Visual Studio Code Configuration for LSST Development
LEGAL
Licensing LSST DM source code and content
Copyrights for LSST DM work and the COPYRIGHT file
USER DOCS
User documentation style guide
IT OVERVIEW
LSST Information Technology Services Committee (ITSC)

Docs » Building the pipelines.lsst.io site locally — Edit on GitHub

Building the pipelines.lsst.io site locally

You can build the full pipelines.lsst.io site on your own computer. Although this is slightly more complicated than building documentation for a single package, it's the best way to ensure that cross-package links work properly.

Alternatively, you can also build and publish a development version of pipelines.lsst.io with the sqre/infrastructure/documenteer Jenkins CI job. That Jenkins job only accepts development branches of the pipelines_lsst_io repository — not development branches of packages. The method described on this page is currently the only way to build documentation for development branches of packages with the full LSST Science Pipelines stack.

Prerequisites

Before starting, you'll need a working `lsst_distrib` installation.

This installation needs to be a recent daily or weekly build so that any in-development packages will compile with the Stack. Working from the tip of the `master` branch is the norm for LSST software development.

Clone and set up the pipelines_lsst_io repository

pipelines_lsst_io is the main documentation repository for the pipelines.lsst.io site. It contains project-wide content, like installation guides and release notes, and also provides the structure for gathering documentation content from individual packages in the LSST Science Pipelines package stack.

Clone the repository:

```
git clone https://github.com/lsst/pipelines_lsst_io
```

Then set up the pipelines_lsst_io package with EUPS:

```
setup -r pipelines_lsst_io
```

Warning

If you've already have packages set up with the **setup** command, you might need to un-setup them with the **unsetup** command before running `setup -r pipelines_lsst_io`.

pipelines_lsst_io acts as a top-level EUPS package, and its table file defines what packages are included in the pipelines.lsst.io documentation site.

Install Documenteer, the documentation tooling

🔗 https://developer.lsst.io/stack/building-single-package-docs.html

Mode 2: Building single package documentation for tight-loop development.

developer.lsst.io

Prerequisites
Install Documenteer, the documentation tooling
Downloading and setting up a package
Building the package's documentation
Deleting built documentation
Further reading
Building the pipelines.lsst.io site locally
Building pipelines.lsst.io with Jenkins (sqre/infra/documenteer)
Testing the LSST DM Stack with the Jenkins stack-os-matrix Job
Coverage Analysis
EUPS Tutorial
The LSST Software Build Tool
Adding a New Package to the Build
Managing license and copyright in Stack packages
Distributing Third-Party Packages with EUPS
Renaming a Stack package
GIT
Git Configuration Standards & Recommendations
Using Git LFS (Large File Storage) for data repositories
EDITORS
Emacs Configuration for LSST Development
SublimeText Configuration for LSST Development
Vim Configuration for LSST Development
Visual Studio Code Configuration for LSST Development
LEGAL
Licensing LSST DM source code and content
Copyrights for LSST DM work and the COPYRIGHT file
USER DOCS
User documentation style guide
IT OVERVIEW
LSST Information Technology Services Committee (ITSC)

Docs » Building single-package documentation locally — Edit on GitHub

Building single-package documentation locally

While developing a package and its documentation, you'll probably want to preview what the documentation looks like when built. The quickest way to do this is with a single-package documentation build.

Keep in mind that a single-package documentation build may have many broken links because package documentation is *meant* to be built in the context of the package stack. To do a full-stack documentation build, see either:

- Building the pipelines.lsst.io site locally
- Building pipelines.lsst.io with Jenkins (sqre/infra/documenteer)

Nevertheless, a single-package build is useful for checking formatting and proof-reading your work. This page describes how to run a single-package documentation build, using pipe_base as the example.

Prerequisites

Before starting this tutorial, you'll need a working `lsst_distrib` installation. This installation should already be set up with a command like **setup lsst_distrib**.

This installation needs to be a recent daily or weekly build since you'll be compiling the pipe_base repository from its `master` branch. Working from the tip of the `master` branch is the norm for LSST software development.

Install Documenteer, the documentation tooling

Documenteer provides tooling to build pipelines.lsst.io. Since it's a PyPI-distributed Python package, you need to install it separately from the EUPS Stack. In a base working directory — not inside a repository directory — get the `requirements.txt` file for pipelines_lsst_io and install it:

```
curl -O https://raw.githubusercontent.com/lsst/pipelines_lsst_io/master/requiremen
pip install -r requirements.txt
```

Do this *after* setting up the EUPS Stack with a command like **setup lsst_distrib**.

If it is not already present in your environment, you may also want to install the graphviz package to enable generation of diagrams:

```
conda install graphviz
```

Note

On a shared resource, like lsst-dev, you will need to use a `--user` flag with **pip**

NSF

U.S. DEPARTMENT OF ENERGY | Office of Science

The doc/ directory of a package

```
doc/
├── SConscript
├── conf.py
├── doxygen.conf.in
├── index.rst
├── lsst.example/
│   ├── index.rst
│   ├── scripts
│   └── tasks
└── manifest.yaml
```

https://developer.lsst.io/stack/module-homepage-topic-type.html

https://developer.lsst.io/stack/argparse-script-topic-type.html

https://developer.lsst.io/stack/task-topic-type.html

Rubin Observatory

Great documentation pages aren't just written, they're also *designed*

Start by asking yourself (or others!) what the user needs from your documentation.

Documentation is an exercise in empathy.

NSF
U.S. DEPARTMENT OF ENERGY | Office of Science

Adding a documentation page

What type of documentation is it?

Tutorials
- Learning-oriented
- For “newcomers” to get started
- A self-contained lesson that the reader performs
- Provides an example

How-to guides
- Goal-oriented
- Shows how to solve a specific problem
- Series of steps

Concept guides
- Understanding-oriented
- Explains
- Provides background & context

Reference guides
- Information-oriented
- Accurate and complete
- Examples:
  - Numpydoc, Doxygen “docstrings”
  - Task reference pages

These types all work together, and ultimately we need to provide all four types

These types can be fractal (an example in a API reference).
Often just link from one page to another (we’re on the web).

Credit: https://www.divio.com/en/blog/documentation/

Elements of a good documentation page

- **Self-contained**

  No previous page, no next page. But lives within the site's web.

- **Specific and limited purpose**

  A topic (i.e., a page) should have a specific, well-defined purpose.

- **Conform to type**

  A type is like a template.

- **Link richly**

  Let the reader forage for information. Think of Wikipedia.

- **Establish context**

  Use the first paragraph to establish the topic's purpose and its relationships to other topics.

- **Assume the reader is qualified**

  Link to introductory topics, don't repeat introductory concepts.

- **Stay on one level**

  Don't evolve level of detail like a narrative.

Based on "Every Page is Page One" by Mark Baker

Adding a documentation page

1. Create a page (rst file) in the module documentation directory or a subdirectory.
2. Add that page to the "toctree" in the parent index.rst file.

```
doc
├── SConscript
├── conf.py
├── doxygen.conf.in
├── index.rst
├── lsst.example
│   ├── index.rst
│   ├── howto-get-butler-dataset.rst
│   ├── scripts
│   └── tasks
└── manifest.yaml
```

doc/lsst.example/index.rst

```
.. py:currentmodule:: lsst.example

.. _lsst.example:

############
lsst.example
############

.. _lsst.example-using:

Using lsst.example
==================

.. toctree::
   :maxdepth: 1

   howto-get-butler-dataset
```

Adding a documentation page

In any documentation page you write, the first paragraph has two roles:

1. State what the reader will learn from the page.
2. Establish the context of this doc, usually by linking to overviews and related docs.

```
.. _howto-get-butler-dataset:

#################################
How to get a dataset with the Butler
#################################

With the LSST Science Pipelines, you don't
directly interact with data files, such as FITS.
Instead, datasets are managed by the Butler in a
repository. This page describes how to get a
dataset using the Butler Python class.

<the actual how-to goes here>
```

link to a conceptual overview page

link to an API reference

Science Pipelines documentation templates

Rubin Observatory

- Templates give the documentation consistency, which is important to readers' success.
- We have a number of templates already, more will be added as we design the types of documentation content we need.

Page templates (topic types in technical writing jargon) are listed in the Developer Guide.

NSF
U.S. DEPARTMENT OF ENERGY | Office of Science

Science Pipelines documentation templates

Rubin Observatory

Direct message @sqrbot-jr in Slack to create a new file from a template.

U.S. DEPARTMENT OF ENERGY | Office of Science

Module homepage topic type

Rubin Observatory

NSF

U.S. DEPARTMENT OF ENERGY | Office of Science

Task topic type

pipelines.lsst.io

LSST Science Pipelines Forum Docs LSST.org →

LSST Science Pipelines » lsst.ip.diffim » NumberSciSourcesMetricTask « previous | next »

Search

On this page

NumberSciSourcesMetricTask
Processing summary
Python API summary
Butler datasets
Input datasets
Retargetable subtasks
Configuration fields
connections
saveMetadata

NumberSciSourcesMetricTask

`NumberSciSourcesMetricTask` computes the number of science sources created when processing data through image differencing (as the `ip_diffim.numSciSources` metric). It requires source catalogs (a `src` dataset) as input, and can operate at image-level or coarser granularity.

Processing summary

`FractionDiaSourcesToSciSourcesMetricTask` reads source catalogs (`src` datasets) and adds up the number of sources in those catalogs.

Python API summary

```
from lsst.ip.diffim.metrics import NumberSciSourcesMetricTask
```

class **NumberSciSourcesMetricTask** (***kwargs*)

Task that computes the number of cataloged science sources...

attribute **config**

Access configuration fields and retargetable subtasks.

method **run** (*sources*)

Count the number of science sources...

See also

See the **NumberSciSourcesMetricTask** API reference for complete details.

Butler datasets

Input datasets

`sources`

The catalog type for science sources (default: `src`).

Retargetable subtasks

No subtasks.

Configuration fields

connections

Data type

`lsst.pipe.base.config.Connections`

Rubin Observatory

NSF

U.S. DEPARTMENT OF ENERGY | Office of Science

Script topic type

pipelines.lsst.io

LSST Science Pipelines Forum Docs LSST.org →

LSST Science Pipelines » lsst.verify » dispatch_verify.py « previous | next »

Search

On this page

dispatch_verify.py
positional arguments
optional arguments
Environment arguments
SQUASH API arguments

dispatch_verify.py

Upload LSST Science Pipelines Verification **Job** datasets to the SQUASH dashboard.

Job JSON files can be created by **lsst.verify.Job.write** or **lsst.verify.output_quantities**. A **Job** dataset consists of metric measurements, associated blobs, and pipeline execution metadata. Individual LSST Science Pipelines tasks typically write separate JSON datasets. This command can collect and combine multiple Job JSON datasets into a single Job upload.

Configuration

dispatch_verify.py is configurable from both the command line and environment variables. See the argument documenation for environment variable equivalents. Command line settings override environment variable configuration.

Metadata and environment

dispatch_verify.py can enrich Verification Job metadata with information from the environment. Currently dispatch_verify.py supports the Jenkins CI and the LSST Data Facility (LDF) execution environments.

In the Jenkins CI execution environment (`--env=jenkins`) the following environment variables are consumed:

- `BUILD_ID` : ID in the CI system
- `BUILD_URL` : CI page with information about the build
- `PRODUCT` : the name of the product built, e.g. 'validate_drp'
- `dataset` : the name of the dataset processed, e.g. 'validation_data_cfht'
- `label` : the name of the platform where it runs

If `--lsstsw` is used, additional Git branch information is included with Science Pipelines package metadata.

In the LSST Data Facility execution environment (`--env=ldf`) the following environment variables are consumed:

- `DATASET` : the name of the dataset processed, e.g 'HSC RC2'
- `DATASET_REPO_URL` : a reference URL with information about the dataset
- `RUN_ID` : ID of the run in the LDF environment
- `RUN_ID_URL` : a reference URL with information about the run
- `VERSION_TAG` : the version tag of the LSST software used, e.g. 'w_2018_18'

Note: currently it is not possible to gather Science Pipelines package metadata in the LDF environment, thus if `--env=ldf` is used `--ignore-lsstsw` is aslo used by default in this environment.

```
usage: dispatch_verify.py [-h] [--test] [--write PATH] [--show]
                          [--ignore-blobs] [--env {jenkins,ldf}]
                          [--lsstsw PATH] [--package-repos [PATH [PATH ...]]]
                          [--ignore-lsstsw] [--url URL] [--user USER]
                          [--password PASSWORD]
                          json [json ...]
```

More information is available at https://pipelines.lsst.io.

positional arguments

Rubin Observatory

U.S. DEPARTMENT OF ENERGY | Office of Science

🔗 https://developer.lsst.io/user-docs/index.html

Rubin Observatory

Writing for users

Your documentation is a tool — a product — that is used by your users.

When you're writing docs, but yourself in the user's shoes to create content that's:

- clear and simple
- easy and fast to consume
- assists users in accomplishing their goals

NSF

U.S. DEPARTMENT OF ENERGY | Office of Science

The style guide is here to help you

In DM, we have primarily adopted the Google Content Style guide: developers.google.com/style/

developers.google.com

Google developer documentation style guide

Search

English

Introduction
Style guide
Style guide highlights
Other editorial resources
What's new

Key resources
Word list
Product names
Text-formatting summary

General principles
Style and tone
Documenting future features
Writing accessible documentation
Writing for a global audience
Writing inclusive documentation
Avoiding excessive claims
Using other sources

Language and grammar
Abbreviations
Active voice
Anthropomorphism
Articles (a, an, the)
Capitalization
Clause order
Contractions
Plurals in parentheses
Possessives
Prepositions
Present tense
Pronouns
Second person
Spelling
Verb forms in reference documentation

Punctuation
Colons
Commas
Dashes
Ellipses
Exclamation points
Hyphens
Parentheses
Periods
Pluralizing a single letter
Quotation marks
Semicolons
Slashes

Home > Products > Style

Rate and review

Highlights

Table of contents

Introduction
Tone and content
Language and grammar
Formatting, punctuation, and organization
Images

Key Point: An overview of some of the highlights of the style guide.

Introduction

The style guide covers a lot of material, so the following page provides an overview of its most important points. For more information about topics on the page, follow the links.

Tone and content

- Be conversational and friendly without being frivolous.
- Don't pre-announce anything in documentation.
- Use descriptive link text.
- Write accessibly.
- Write for a global audience.

Language and grammar

- Use second person: "you" rather than "we."
- Use active voice: make clear who's performing the action.
- Use standard American spelling and punctuation.
- Put conditional clauses before instructions, not after.
- For usage and spelling of specific words, see the word list.

Formatting, punctuation, and organization

- Use sentence case for document titles and section headings.

Design your text to be read on the web

Rubin Observatory

- Users won't read your text thoroughly in a word-by-word manner.
- The first two paragraphs must state the most important information.
- Start subheads, paragraphs, and bullet points with information-carrying words.

From: F-Shaped Pattern For Reading Web Content by Jakob Nielsen, http://ls.st/fzp.

NSF
U.S. DEPARTMENT OF ENERGY | Office of Science

Writing for the web

- Most people don’t read the web sentence-by-sentence.
- Use lots of meaningful headers.
- 2–3 sentence paragraphs.
- First one or two words of a paragraph should convey meaning.
- Use lists and typographic elements for structure.
- Use code samples.

Tone: be conversational without being frivolous

Rubin Observatory

❌ Please note that completion of the task requires the following prerequisite: executing an automated memory management function.

❌ Then—BOOM—just garbage-collect (or *collecter des garbáge*, as they say in French), and you're golden.

✅ To clean up, call the `collectGarbage()` method.

The most straightforward sentence is often the best sentence.

Avoid sounding "clever."

Avoid jokes unless you know what you're doing.

Contractions are fine.

From https://developers.google.com/style/tone

NSF
U.S. DEPARTMENT OF ENERGY | Office of Science

Use second person

Rubin Observatory

❌ In the previous tutorial in the series we used processCcd.py to calibrate a set of raw Hyper Suprime-Cam images. Now we'll learn how to use the LSST Science Pipelines to inspect processCcd.py's outputs by displaying images and source catalogs in the DS9 image viewer. In doing so, we'll be introduced to some of the LSST Science Pipelines' Python APIs.

✅ In the previous tutorial in the series you used processCcd.py to calibrate a set of raw Hyper Suprime-Cam images. Now you'll learn how to use the LSST Science Pipelines to inspect processCcd.py's outputs by displaying images and source catalogs in the DS9 image viewer. In doing so, you'll be introduced to some of the LSST Science Pipelines' Python APIs.

From https://developers.google.com/style/person

Use second person: speak directly to the user

Rubin Observatory

❌ When setting up a product, a user may specify a version, or accept the current default.

✅ When setting up a product, you may specify a version, or accept the current default.

User documentation must be actionable for the user.

From https://developers.google.com/style/person

NSF
U.S. DEPARTMENT OF ENERGY | Office of Science

Use the active voice

Rubin Observatory

❌ The service is queried, and an acknowledgment is sent.

✅ Send a query to the service. The server sends an acknowledgment.

Passive? If you can add "by monkeys" to a sentence, it's passive.

Active voice makes it clear who performs the action, which makes for more understandable documentation.

From https://developers.google.com/style/voice

NSF
U.S. DEPARTMENT OF ENERGY | Office of Science

Use the passive voice when the actor is unimportant to the user

Rubin Observatory

- ✅ The database was purged in January.
- ✅ Over 50 conflicts were found in the file.

From https://developers.google.com/style/voice

NSF
U.S. DEPARTMENT OF ENERGY | Office of Science

Write in the present tense

Rubin Observatory

❌ Send a query to the service. The server will send an acknowledgment.

✅ Send a query to the service. The server sends an acknowledgment.

❌ You can send an unsubscribe message. The server would then remove you from the mailing list.

✅ If you send an unsubscribe message, the server removes you from the mailing list.

https://developers.google.com/style/tense

Don't anthropomorphize software and hardware systems

Rubin Observatory

❌ The `doApCorr` configuration field tells the `ForcedPhotCcdTask` whether to apply aperture corrections.

✅ Set the `doApCorr` configuration field to `True` to enable aperture corrections.

Software doesn't:

- **tell**
- **know**
- **want**
- **...**

See https://developers.google.com/style/anthropomorphism

NSF
U.S. DEPARTMENT OF ENERGY | Office of Science

Avoid ableist language and write inclusively

Rubin Observatory

❌ Replace the dummy variable in this example with the appropriate variable.

✅ Replace the placeholder variable in this example with the appropriate variable.

🔗 https://developers.google.com/style/inclusive-documentation

NSF
U.S. DEPARTMENT OF ENERGY | Office of Science

Use sentence case for section headlines

Rubin Observatory

❌ LSST-Related Notebook Repositories

✅ LSST-related notebook repositories

❌ Getting Notebooks from GitHub

✅ Getting notebooks from GitHub

https://developers.google.com/style/headings

NSF
U.S. DEPARTMENT OF ENERGY | Office of Science

Introducing a list

Rubin Observatory

❌ Use the LSST Science Pipelines to:

- do forced photometry
- measure galaxy shapes
- apply photometric and astrometric calibrations

✅ You can use use the LSST Science Pipelines any of the following purposes:

- To perform forced photometry of a set of images.
- To measure the shapes of galaxies.
- To apply photometric and astrometric calibrations.

The introductory sentence must be complete. Don't complete it with list items.

See https://developers.google.com/style/lists

NSF
U.S. DEPARTMENT OF ENERGY | Office of Science

List tips

✓ Use parallel structure.

✓ Add a period if the item is a complete sentence.

✓ Don't add add punctuation if the item is a single word.

✓ Use the appropriate type of list:

- Bullet lists for unordered items
- Numbered lists for procedures.
- Definition lists to pair terms with definitions or explanations.

See https://developers.google.com/style/lists

Introducing a code sample

Rubin Observatory

❌ Run:

```
setup lsst_distrib
```

to set up the LSST Science Pipelines.

✅ Run this command to set up the LSST Science Pipelines:

```
setup lsst_distrib
```


NSF
U.S. DEPARTMENT OF ENERGY | Office of Science

Technical writing is a skill, but *you can learn it*

Rubin Observatory

Spend time with the style guide (and have it open while writing)

https://developers.google.com/style

Critically analyze docs from major projects/companies you use

https://docs.github.com

Books

Docs for Developers: An Engineer's Field Guide to Technical Writing (2021) Bhatti et al
Every page is page one (2013) Baker
... search for "technical writing" or "technical communication"

Feedback from peers and users

NSF
U.S. DEPARTMENT OF ENERGY | Office of Science

Rubin Observatory

Whew, that was a lot to learn.

Don't worry, everything is documented and you can ask for help on Slack.

NSF

U.S. DEPARTMENT OF ENERGY | Office of Science

Resources for learning more

Slack

#dm-docs-support channel for help writing docs

User documentation style guide

https://developer.lsst.io/user-docs/index.html

ReStructuredText style guide

https://developer.lsst.io/restructuredtext/style.html

Docstrings

https://developer.lsst.io/python/numpydoc.html

Stack developer guide (including writing docs for pipelines.lsst.io)

https://developer.lsst.io/index.html#dm-stack

Writing technotes

https://developer.lsst.io/project-docs/technotes.html

Documentation portal

https://www.lsst.io