Science Pipelines Doc Sprint 2016-12

Date

07 Dec 2016

Attendees

- Jonathan Sick
- Simon Krughoff
- John Swinbank
- Unknown User (mssgill)

Goals

This is a design sprint to kick off the LSST Science Pipelines documentation reboot. Our goal is to a create a tangible vision of what the Science Pipelines documentation will be. Questions we want to answer are:

- Who are the users of Science Pipelines documentation? What does each group want to get out of the Science Pipelines and its documentation? Do those needs conflict? Do we need to prioritize one user group in the initial implementation?
- What are the boundaries of the Science Pipelines documentation (the site at https://pipelines.lsst.io)? What are adjacent documentation projects that the Science Pipelines documentation might link against?
- What's the curriculum for learning the Science Pipelines? What are the concepts that the Science Pipelines documentation site needs to cover? How are these concepts organized (hierarchically or as a bottom-up information network). Do different types of users need specific entry points into the documentation and Science Pipelines itself?
- What kinds of content are we going to be producing? What do the templates of these topic types look likes?
- How are concepts unique to Science Pipelines, like tasks and command line tasks, documented in both a code and information architecture sense?

Intended Sprint Products

These are suggested products and outcomes from the sprint:

- **A map of the science pipelines site.** This map should resolve into individual HTML/reStructuredText documents (topics in *Every Page is Page One* terminology). Each topic should be annotated with:
  - Topic name.
  - Content purpose and scope.
  - Topic type (i.e., template).
  - Adjacent topics (topics that link into this page; topics this page will link out to).
- **Topic types and templates.** Each template shapes how different types of topics are written. Examples can be: API reference, task, command line task, tutorial project, conceptual overview, recipe. See *Every Page is Page One* Chapter 9: EPPO Topics Conform to a Type.
- **Timelines**. Timeline for content and for **documentation infrastructure**.

Prep Work/Background Reading Material

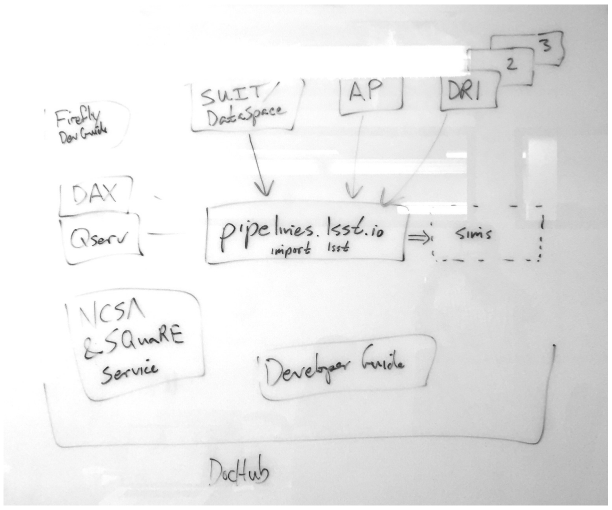
- Read Every Page is Page One.
- Jonathan Sick's "Pipelines Documentation Site Organization Sketch" is on clo.
- LDM-493: Data Management Documentation Architecture.
- Potentially relevant design docs, which may be cross-referenced with or otherwise relate to Science Pipelines docs:
  - LDM-151 (DM Applications Design)
  - LSE-163 (Data Products Definition Document)
- validate-base documentation
- Astropy documentation

Meeting Logistics

- Tuesday December 6: campfire chat at Bentley's or elsewhere.
- Wednesday December 7. 9:00 am to 5:00 pm. LSST Workroom.
- Thursday December 8. 9:00 am to 5:00 pm. LSST Workroom.
- Friday December 9. 9:00 am to 5:00 pm (or as participants depart). LSST Workroom.

Discussion items

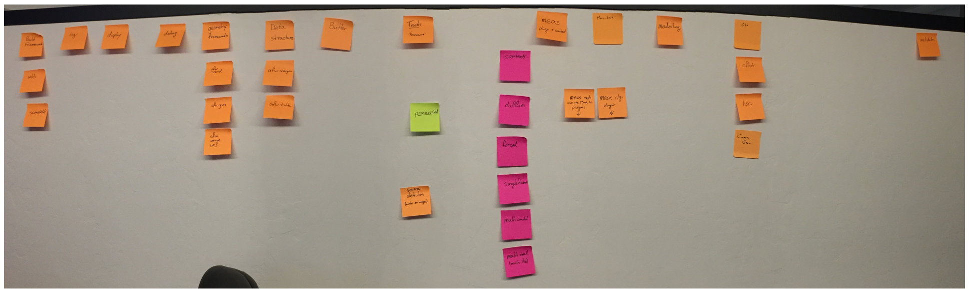
Time	Item	Who	Notes

	What is the scope of the "Science Pipelines" documentation site? • Note that important obs packages are outside lsst_distrib; omitting the obs packages would reduce the current usability of the Pipelines documentation.		• Technical constraint: tightly coupled packages should be documented together since docs will be versioned tightly with the codebase (docs embedded in Git; also known as 'docs as code'). This is an LSST the Docs feature: https://sqr-006.lsst.io. • We agreed that a lot of middleware (things beyond Princeton and UW) should be included in pipelines.lsst.io because of the tight API integration, including: ◦ task/supertask framework ◦ butler ◦ logging ◦ display packages • Example: document the Butler API in pipelines.lsst.io, but document the DAX service elsewhere. ◦ Butler is an API that has implementations for different backends. ◦ Document implementations to each backend. ◦ Doc how to write an implementation. • Example: document the Firefly display package in pipelines.lsst.io, but document Firefly itself elsewhere. • There is a list of obs packages that will be supported. These will be included in pipelines.lsst.io. • lsst.validate packages will be in pipelines.lsst.io. • Can all packages in the lsst Python namespace be thought of as pipelines.lsst.io (excluding simulations).? Is pipelines.lsst.io effectively the documentation for the "lsst" python package? • Think of pipelines.lsst.io as documentation for the open source project that might be used in other contexts besides LSST AP and DRP pipelines (other observatories, building L3 data products). Data release documentation will specify exactly how the Science Pipelines were used to build a data release.  Firefly Dev Guide SUIT/DataSpace AP DRI 2 3 DAX Qserv pipelines.lsst.io import lsst Sims NCSA & SQuaRE Service Developer Guide DocHub
	Boundary between Pipelines docs and the Developer Guide Should the pipelines documentation cover developer and build-oriented topics currently in the DM Developer Guide? Do pipelines users need to be able to create Stack packages to make Level 3 data products? • lsstsw and lsst-build • Structure of Stack packages (including sconsUtils and EUPS details) • etc?		• developer.lsst.io is intended to define policies and practices specific to DM staff. We can't use it as documentation to end users. • If the build and packaging system are described in pipelines.lsst.io, it could be awkward for other software projects, like Qserv and Sims, that also depend on EUPS/sconsUtils/lsst-build/lsstsw, etc.. • However, putting build/packaging documentation in pipelines.lsst.io probably makes the most sense for astronomers extending the stack. pipelines.lsst.io is already where astronomers will look to learn how to write new packages against the Pipelines API. Overall, we can just learn that pipelines.lsst.io is where build and packaging is fundamentally documented.
	Science Pipelines docs and LDM-151		• LDM-151 is where we're designing and planning the stack. • Eventually it will grow to say what the Stack **is.** • pipelines.lsst.io will also say what the Stack is. • LDM-151 is change controlled: not continuously deployed like the Stack documentation. • What if LDM-151 is kept as a record of the Stack used for reviews and related communities? And most users only use pipelines.lsst.io? • This needs to be discussed by DM/TCT leadership. • Existing proposal: https://ldm-493.lsst.io/v/v1/index.html#change-controlled-design-documents (suggests that content is transplanted and single sourced in design docs).

	Who are our users? • What user group should be prioritized? • What are common activities that this group wants to achieve? What documentation will assist with that? • Where do the needs of different groups overlap?		• DM developers in construction ◦ Need API references most. ◦ Currently learn APIs by introspection or reading the source and code that uses an API. Doxygen isn't useful. ◦ Descriptions of how tasks fit together (both API-wise, and higher-level concepts; even LDM-151-level). ◦ Examples to help us develop one package given lower level APIs. ◦ Run tasks for validating processing; run on verification clusters. ◦ DM is the biggest consumer of pipelines.lsst.io. • Construction-era science collaborations (sims users?) ◦ Currently consumers of Sims (MAF). ◦ Many won't contribute to the pipelines stack. ◦ May want to give feedback. Need algorithmic descriptions. • DESC ◦ Running real imaging data now with the stack ◦ Want to contribute feedback (knowledge). E.g. on algorithms. ◦ Want to contribute packages. E.g. twinkles. ◦ Want to implement a measurement algorithm and compare against the performance of factory algorithms. ◦ Need: ▪ developer docs (to support development) ▪ algorithm background (to comment on) ▪ how to run pipelines on their own infrastructure. • LSST operators/scientists in operations ◦ DRP may want an internal ops guide (out of scope) ◦ Science directorate will have similar needs to DM developers now. • 'DataSpace' users in operations ◦ SDSS experience: Small queries to subset data. Complex queries to get objects of interest. Use cut-out service to give context to catalogs. ◦ Will want to run tasks on a subset of image data. Customize our algorithms. ◦ Use Butler to get/put datasets within their storage quota ❓. ◦ Develop and test algorithms that may be proposed for incorporation in DRP. • Other observatories/surveys Summary • DM developer needs generally match the needs of all other groups, possibly with the exception of some conceptual framing documentation. DM will be API oriented, whereas new users will need more conceptual docs. • Need priorities, still.
	EUPS Packages as units of organization • It's natural to organize documentation (to some extent) according to units of EUPS packages, given that doc content should live with code. Should every EUPS package have a topic page and be linked from the homepage (like the astropy docs do for sub-packages)? Are there exceptions where documentation that may live in an EUPS package should actually be organized altogether independently of EUPS package structure? • What should typical in-package documentation look like? See https://validate-drp.lsst.io as a prototype, and https://docs.astropy.org in general. • To what extent should documentation refer to EUPS packages (e.g., afw) versus Python namespaces (lsst.afw)?		• Document at the level of the Python module. e.g. afw.image, afw.table, pipe.base, not necessarily at the Git repository level. • Docs live inside packages and package docs can be built locally and independently of the full pipelines.lsst.io site. • However, the pipelines.lsst.io homepage can arrange docs for modules into topical groupings.

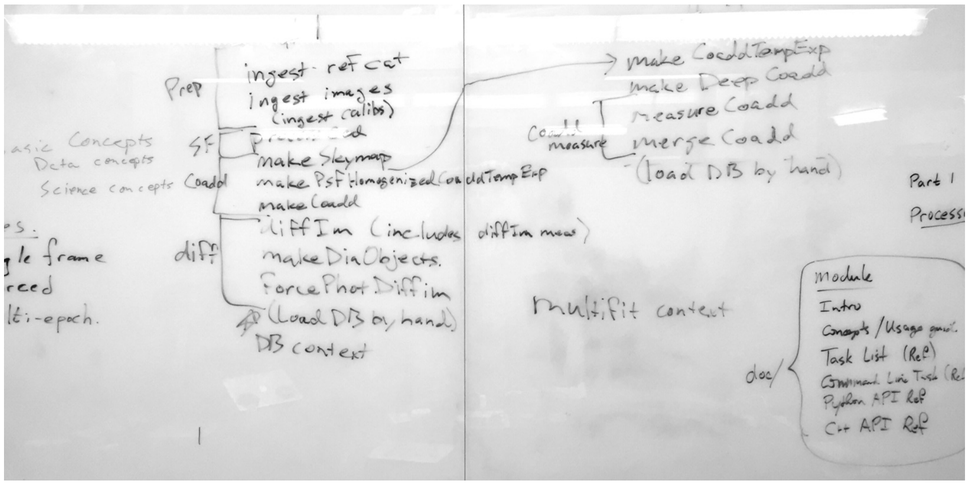
What is the structure of the documentation homepage?

- The homepage is important for orienting users. The structure of the homepage should present a coherent vision for what the Science Pipelines are and how they're used.
- See also https://community.lsst.org/t/pipelines-documentation-site-organization-sketch/1088/1 and LDM-151

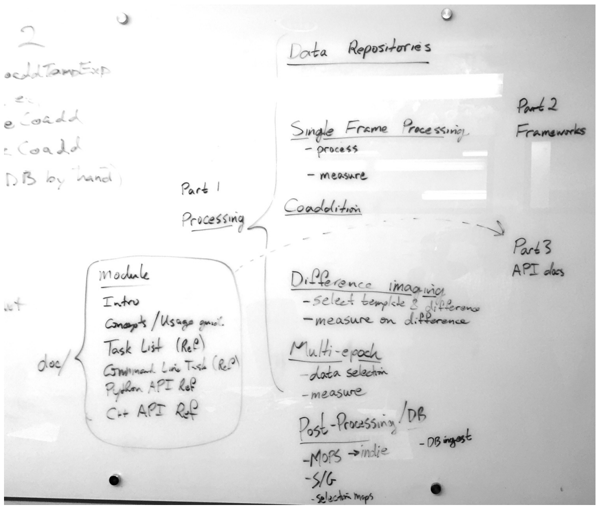


Frameworks.

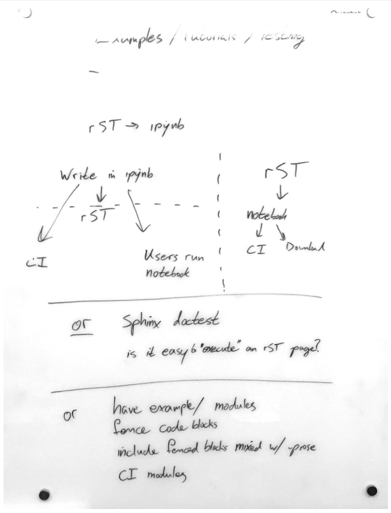
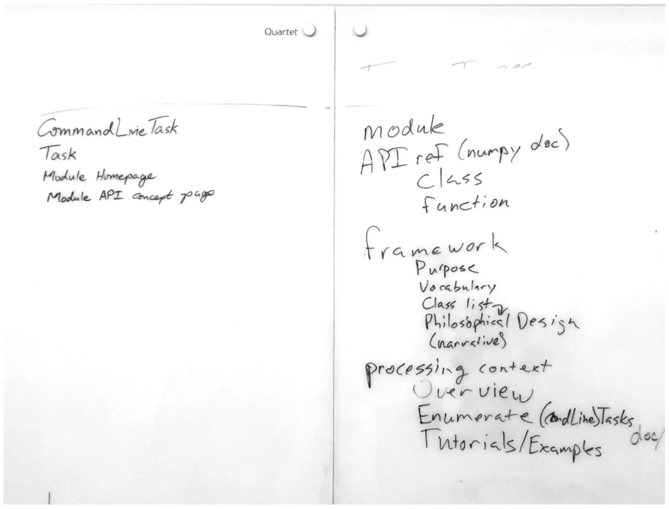
- obs
- meas.
- modelling.
- tasks.
- Butler/Data Access Framework.
- Data structures
- geometry
- display
- log
- debug
- validate
- Build system



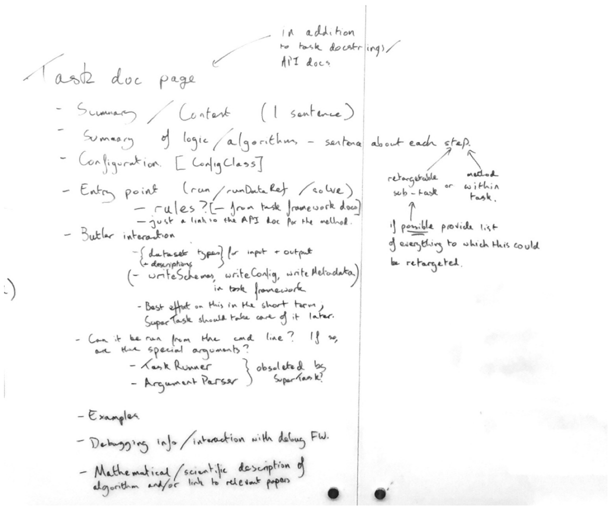
Twinkles workflow.



Homepage structure.

	Where should concepts of science interest (such as algorithm details) be documented? • Docstrings of code that implements algorithms? • Tasks/Command line task interface references? • Concept topics that then introduce task /API references? • To what extent are LSST design documents (e.g., LDM-151) cross-linked and referenced?		• Algorithms don't match Python/C++ APIs 1:1. Indicates that algorithms should be described at a higher level. • Tasks might be the best home for algorithm documentation. • Need for higher-level overviews that describe "processing topics" that link to composed tasks.
	How should examples and tutorials be produced? • All tutorial and in-text examples need to be runnable. • How do we leverage the example/ directories? • Should documentation pages essentially be written as Jupyter notebooks?		 Examples / Tutorials / Testing rST → ipynb Write in ipynb rST CI Users run notebook rST notebook CI Download or Sphinx doctest is it easy to "execute" an rST page? or have example/ modules fence code blocks include fenced blocks mixed w/ prose CI modules We need additional prototyping and design discussion before we identify a pattern for producing and testing examples in documentation.
	How should C++/Python API reference documentation be produced? • See https://community.lsst.org/t/documentation-and-links-for-python-wrapped-c-code-in-sphinx/1392		• There should be a small discussion between the pybind11 transition team and SQuaRE doc engineering to design and choose a system.
	Listing topic types and templates • What are all the distinct *types* of things we'll need to document? • What should each type of content look like?		 Quartet CommandLineTask Task Module Homepage Module API concept page module API ref (numpy doc) Class function framework Purpose Vocabulary Class list Philosophical Design (narrative) processing context Overview Enumerate (CmdLine)Tasks Tutorials/Examples doc/

Preliminary listing.



Task doc page

in addition to task docstrings / API docs

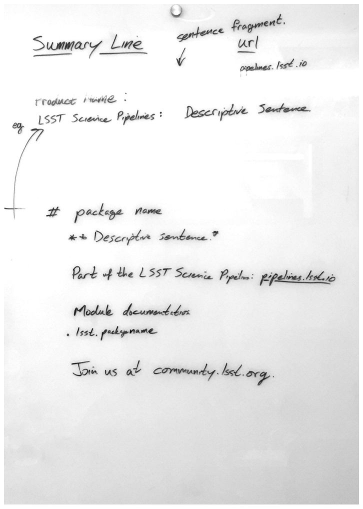
- Summary / Context (1 sentence)
- Summary of logic / algorithms - sentence about each step.
- Configuration. [ConfigClass]
- Entry point (run / runDataRef / solve)
  - rules? [- from task framework docs]
  - just a link to the API doc for the method.
- Butler interaction
  - {dataset types + descriptions} for input + output
  - (- writeSchemas, writeConfig, writeMetadata) in task framework
  - Best effort on this in the short term, SuperTask should take care of it later.
- Can it be run from the cmd line? If so, are the special arguments?
  - Task Runner
  - Argument Parser

  } obsoleted by SuperTask?
- Examples
- Debugging info / interaction with debug FW.
- Mathematical / scientific description of algorithm and/or link to relevant papers

retargetable sub-task or method within task.

if possible provide list of everything to which this could be retargeted.

Task topic type.



Summary Line

sentence fragment.

url

pipelines.lsst.io

Product Name :

eg. LSST Science Pipelines : Descriptive Sentence.

# package name

** Descriptive sentence.*

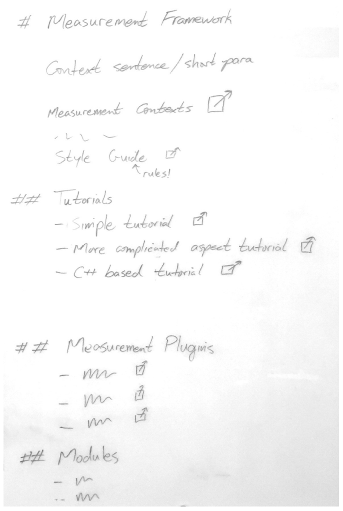
Part of the LSST Science Pipelines: pipelines.lsst.io

Module documentation

- lsst.packagename

Join us at community.lsst.org.

README topic type + GitHub summary line.



# Measurement Framework

Context sentence / short para

Measurement Contexts

Style Guide

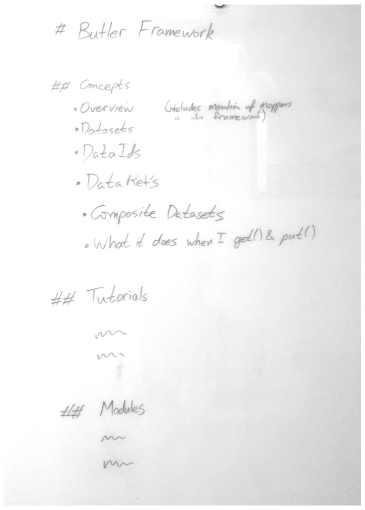
rules!

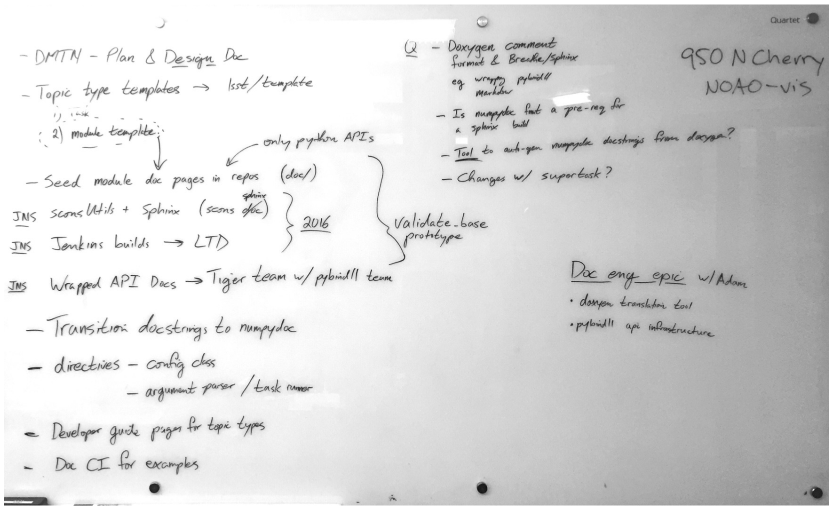
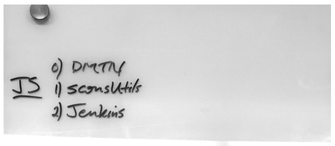
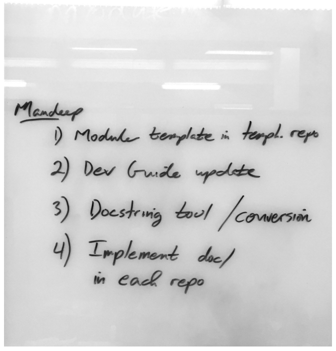
## Tutorials

- Simple tutorial
- More complicated aspect tutorial
- C++ based tutorial

## Measurement Plugins

## Modules

			Measurement framework topic example  # Butler Framework ## Concepts • Overview (includes mention of mappers in the framework) • Datasets • DataIds • DataRef's • Composite Datasets • What it does when I get() & put() ## Tutorials ## Modules Butler framework topic example.
	Community.lsst.org and the docs		• Approach 1: use community.lsst.org as a draft for docs: see new content on the forum, write the docs, and then post a link to that doc in the original topic. This is a culture and process problem. • Approach 2: auto-link to community.lsst.org topics from documentation pages. Can be done by looking for Community topics that link to the documentation site, and by looking for certain watch words that are embedded in the metadata of each reStructuredText page. DocEng will make this.
	Tagging command line tasks		We'll have lots of lists of command line tasks in two places: module topic pages and in processing context sections of the home page. • On the homepage we'll want to curate topical groups. Given the small number of command line entrypoints this can be maintained manually. Eventually we can add tag metadata to each task to support auto-generated lists • On module pages the command line task list can be alphabetical.
	Task configuration and re-targetting		
	Command line task topic types vs task topic types		Task framework documentation should document the philosophy of tasks vs command line tasks • One stance is that command line tasks are aggregations of tasks. The tasks are what contains algorithms, and is where the algorithm should be documented. • However we discussed that the difference may not be meaningful and that tasks and command line tasks should be documented together in a single topic type.
	Measurement extensions listing		We can look at the registry of measurement plugins (extensions)
	Important frameworks		Important/interesting frameworks are the ones that span multiple modules • Butler • measurement • tasks

	Implementation plan		 - DMTN - Plan & Design Doc - Topic type templates → lsst/template 1) task 2) module template only python APIs - Seed module doc pages in repos (doc/) JNS sconsUtils + Sphinx (scons ~~doc~~ sphinx) JNS Jenkins builds → LTD 2016 validate_base prototype JNS Wrapped API Docs → Tiger team w/ pybind11 team - Transition docstrings to numpydoc - directives - config class - argument parser / task runner - Developer guide pages for topic types - Doc CI for examples Q - Doxygen comment format & Breathe/sphinx eg wrapping pybind11 markdown - Is numpydoc fmt a pre-req for a sphinx build - Tool to auto-gen numpydoc docstrings from doxygen? - Changes w/ supertask? 950 N Cherry NOAO-vis Doc eng epic w/Adam • doxygen translation tool • pybind11 api infrastructure Quartet  JS 0) DMTN 1) sconsUtils 2) Jenkins  Mandeep 1) Module template in templ. repo 2) Dev Guide update 3) Docstring tool / conversion 4) Implement doc/ in each repo

Homepage Outline

LSST Science Pipelines

- Installation and setting up
- Processing data: a tutorial
- Release Notes
- Community, and getting help
- How to report issues
- How to contribute

Processing Data

In the beginning, this will be a single page that describes each measurement context and the main processing tasks that are done here.

****

The Processing Data section is **oriented around command line entrypoints (command line tasks or supertasks)** and documents processing patterns and algorithmic considerations.

The sections are patterned around typical user pipelines and processing/measurement **contexts** (single frames, coadds, difference imaging, and multi-epoch datasets). Contexts are slightly different from LDM-151 Section 5 headers. For example, we treat coaddition and difference imaging as different contexts.

In each section, there will be:

- Overview pages that provide a narrative to command line processing and algorithms.
- Tutorials that illustrative command line tasks with realistic datasets.
- Lists of command line tasks, linking to their reference pages. Command line task reference pages are hosted inside package documentatation. Command line task reference pages also link to **task** reference pages. Organize command line tasks between:
    - processing data
    - measuring data

Data ingest

- Overview
- tutorials

Single Frame

- Overview — *what do we do in a single frame context. Then link to processCcd.*
- Tutorials

Coaddition

- Overview topic
- Tutorials

Difference imaging

- Overview
- Tutorial

Multi-epoch object characterization

- Overview. E.g. https://lsst-web.ncsa.illinois.edu/doxygen/x_masterDoxyDoc/pipe_tasks_multi_band.html
- Tutorial

Postprocessing

- Overview(s)
- tutorials
- May need finer grained organization

Frameworks

- Measurement framework
- Butler framework
- task framework
- obs framework
- modelling framework
- geometry framework
- validation framework
- Build, packaging and utility framework

API modules

- lsst.afw.image - Image data structures
- ...

Module topic

lsst.module.name — Readable name

Context establishment paragraph.

Links to related modules, framework pages, and disambiguation.

Design/High Level Overview

If necessary?

Tasks

- Listing of tasks (autogenerated; alphabetical)

Using the lsst.module.name API

- Links to API concept pages
- If it has a C++/Python API

Python API reference

- list of API object reference pages

C++ API reference

- list of API object references pages

Packaging

- Link to EUPS package/GitHub repository
- Dependencies: auto-generated graph/list of EUPS dependencies

Related documentation

- Linked design documents
- Linked technotes
- Linked papers
- Linked Community conversations

Task topic type

TaskName

Summary/context (1 sentence).

Summary of logic/algorith in a paragaph and/or bullet list. Include a sentence about each step, which can be either a) retargetable sub-task, or b) method within task.

Configuration

- Document fields in associated config class
- For subtasks, provide list of everything to which this could be retargeted.

Entrypoint

- Link to API page for the "run" method

Butler Inputs

- dataset type + description of Butler gets()
- Best effort for now; hopefully auto-doc'd in SuperTask framework

Butler Outputs

- dataset type + description of Butler puts()
- Best effort for now; hopefully auto-doc'd in SuperTask framework

Examples

- self-contained example of using this task that can be tested

Debugging

- Debugging framework hooks

Algorithm details

- Extended description with mathematical details

Measurement framework (example of a framework topic page)

Measurement Framework

Context sentence/short paragraph

Framework concepts

- Overview
- Measurement contexts
- ...
- Style guide (rules for creating measurement plugins)

Tutorials

- Simple tutorial for creating a measurement plugin.
- Another tutorial with a more complicated aspect tutorial.
- C++ based tutorial
- ...

Measurement plugins

- a measurement plugin; linking to its class API
- ...

Modules

- list of modules that build up the framework
- ...

Butler framework (example of a framework topic type)

Butler framework

Context sentence/short paragraph.

Framework concepts

- Overview
- Datasets
- DataIds
- Composite Datasets
- What it does when I get() and put()
    - (there might be need for some concept pages that dive into internals)

Tutorials

- ...

Modules

- ...

README topic type

package_name

Desciption from summary line in bold weight.

This package is part of the LSST Science Pipelines: https://pipelines.lsst.io.

Join us at https://community.lsst.org.

Module Documentation

- module homepage link
- for each module in the package

GitHub summary line topic type

LSST Science Pipelines: Descriptive sentence.

https://pipelines.lsst.io

Command line tasks

- IngestCatalogTask
- IsrTask
- MeasurementDebuggerTask
- ProcessImageForcedTask
- DeblendAndMeasureTask
- BaseMeasureTask
- DumpTaskMetadataTask
- ReportImagesInPatchTask
- ReportImagesToCoaddTask
- ReportPatchesTask
- ReportTaskTimingTask
- Generating coadds:
    - AssembleCoaddTask
    - SafeClipAssembleCoaddTask
- GetRepositoryDataTask
- ImageDifferenceTask
- MakeDiscreteSkyMapTask
- MakeSkyMapTask
- MockCoaddTask
- Multi-band processing:
    - DetectCoaddSourcesTask
    - MergeSourcesTask
    - MeasureMergedCoaddSourcesTask
- ProcessImageTask
- RunTransformTaskBase
- CoaddAnalysisTask
- CompareAnalysisTask
- ColorAnalysisTask
- ctrl_pool middleware tasks:
    - BatchCmdLineTask
    - BatchPoolTask
- ProcessCcdTask

More things to discuss/design

- Task list topic types
- Tutorials
- Troubleshooting (when something goes wrong). -> integrate into task lists, and into task reference pages.

Engineering needs

- Turn pipelines_docs into an EUPS package so it can use lsst.utils.getPackageDir rather than assuming that packages are in lsstsw
- Integrate doc builds with sconsUtils

- Branch dashboard pages

Action items

- [ ]