

# Modern documentation tools and approaches

ATLAS Software and Computing Documentation Workshop

@GiordonStark and @MarioLassnig  
December 11th, 2017

<https://indico.cern.ch/event/681173/>

# Outline for today

## Community input

A summary of solicited input from ATLAS community at-large

- What do we use?
- What do we like?
- What do we dislike?

## Available tools

What are some modern tools available used elsewhere?

- How do these tools map to what the community uses?

## Moving forward

Can we move forward and use more centralized tools?

- How might we migrate to these tools?

# What are the main challenges?

## Challenge 1

### **Vague formats**

Good documentation tools should be intuitive and resilient.

Users will not want to write documentation if they don't understand the format or how to make it.

## Challenge 2

### **Out-of-date docs**

Documentation often becomes stale or crusty.

Implementation is easy but maintenance is hard. How do we ensure documentation is up-to-date?

## Challenge 3

### **Lack of good search**

Above all, docs must be accessible. Users should be able to search. If you have to bookmark pages because you can't find it otherwise, then discovery needs to be improved.

# Input

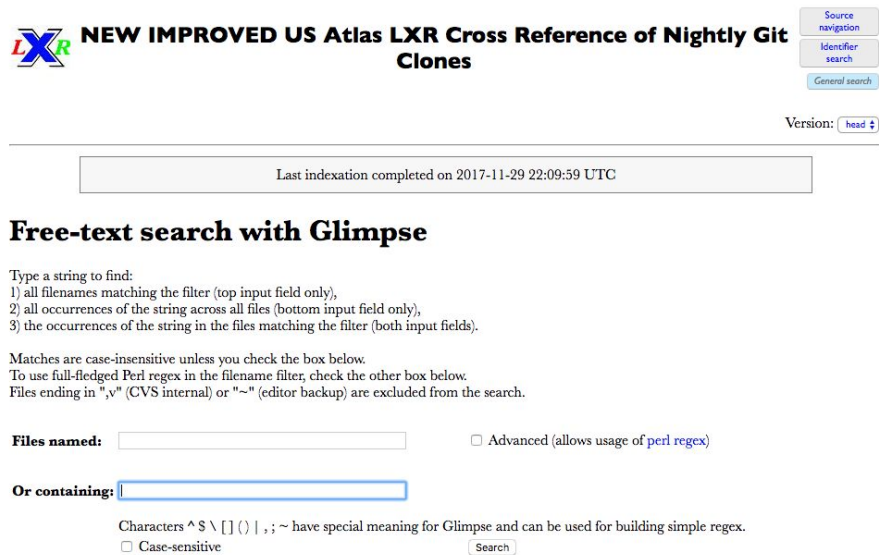
from the ATLAS community

An email was sent out to solicit responses from the ATLAS community about their comments/suggestions

---

# Existing tools

- LXR
  - USATLAS has asked BNL to continue to maintain it as there is no modern tool with equivalent functionality
  - **Can we replace LXR with GitLab search? Continue to support LXR?**
- Twiki
  - Must preserve twikis (no matter what?)
  - Mix of outdated and up-to-date docs
  - Comments about not being proactive and making changes to twiki - it doesn't feel open? Difficult for external people to join.
  - Can't find anything
  - Cannot draw diagrams!
  - ADC: using more and more Google Docs



**LXR** NEW IMPROVED US Atlas LXR Cross Reference of Nightly Git Clones

Source navigation  
Identifier search  
General search

Version: [head](#)

Last indexation completed on 2017-11-29 22:09:59 UTC

### Free-text search with Glimpse

Type a string to find:  
1) all filenames matching the filter (top input field only),  
2) all occurrences of the string across all files (bottom input field only),  
3) the occurrences of the string in the files matching the filter (both input fields).

Matches are case-insensitive unless you check the box below.  
To use full-fledged Perl regex in the filename filter, check the other box below.  
Files ending in ".v" (CVS internal) or "~" (editor backup) are excluded from the search.

Files named:   Advanced (allows usage of [perl regex](#))

Or containing:

Characters ^ \$ \ [ ] ( ) | ; ~ have special meaning for Glimpse and can be used for building simple regex.

Case-sensitive

# How are we currently documenting?

- XRootD
  - Maintained in a Microsoft Word Doc
  - Compiled into PDF and HTML and hosted online
  - Documenting features before implementing them
- Rucio
  - Auto generated from Sphinx to ReadTheDocs: <https://rucio.readthedocs.io/>
  - Rucio-related plugin [Xcache](#) is on Github wiki
  - Modernization of website: <https://rucio.cern.ch/>
- Pilot 2
  - Will be generated using Sphinx and hosted on Github
  - Internal documentation in twiki, technical documents in Google Docs
- Harvester
  - Uses GitHub wiki: <https://github.com/PanDAWMS/panda-harvester/wiki>
- CP Groups and Analyzers
  - Most use Twiki, but not a lot of input from them for this talk
  - Do they want other options for documenting?
  - ATLAS Offline Software uses Jekyll: <https://atlassoftwaredocs.web.cern.ch/>



# Common complaints

- Documentation is all over the place, need to know *what is where*
- Sometimes duplicate documents with minor/major different contents
- No easy way to distinguish current and outdated documents
- Lots of information can only be found in presentations
- Twikis would be fine if automatic deployment to twikis were possible
  - Still, can be considered a heavy case of Stockholm syndrome
- Collaborative editing like Google Docs is important for ADC
- Don't know how to quickly find anything in twiki except for bookmarks
- Would like to have documentation be user-contributed, but it creates a disjointed document because of the many writing styles

# Some questions

- Different documentation for different users: developers, ops, shifters, analyzers - **who maintains each kind?**
- Are we using the best tools?
- Can we retire LXR in favor of GitLab?
- Is it possible to put together an ATLAS style guide for documentation?



# Some suggestions

- Host a **docathon** - a week where everyone is strongly encouraged to make changes only to documentation, update, and do some housekeeping
  - Berkeley Institute for Data Science has done this in the past:  
<https://www.eventbrite.com/e/bids-docathon-kickoff-tickets-32302896834?aff=mcivte>
  - Perhaps would be nice to do this during the (E)YETS or during other breaks
- Doxygen has always been helpful, more effort to have doxygen used consistently
- Create a documentation “style guide” for ATLAS specifying best procedures along with working examples of implementation
- Create a dedicated working group whose job is to maintain documentation
  - Keep docs organized, centrally located, and up-to-date
  - Contact relevant persons to gather needed information
  - Experts on tools and techniques for doc migration

# Available tools

# Documenting code

- C++
  - Doxygen is really the only solution here, but it's a great solution
  - Requires a specific format for your comments
- Python
  - Using docstrings in python, documentation can be extracted with [autodoc](#) (part of [Sphinx](#))
  - Also natively allows for doctesting which can keep documentation fresh automatically
- C++ and Python in the same project
  - [Breathe](#) is a great extension that lets you run doxygen on your C++ code and combine it with Sphinx on your Python code to produce a single set of documentation incorporating both

At the end of the day, users need to be vigilant about adding comments to their code. A central ATLAS style guide for formatting comments will simplify the rest of the process.

# Creating documentation

- Doxygen: can create an HTML output for you
  - ATLAS uses it: <https://atlas-sw-doxygen.web.cern.ch/atlas-sw-doxygen/index.php>
- GitLab pages: host static websites from your repository
  - Many example projects here <https://gitlab.com/groups/pages> including hugo, jekyll, hyde, nanoc, doxygen, gitbook, sphinx, and mkdocs
  - (The same is basically true for GitHub)
  - Our GitLab CERN instance has GitLab pages disabled, in favor of CERN espaces
    - Vendor lock-in to Microsoft Sharepoint
    - Not a good impression for the open-source community...

# Case Studies - Real World Use

# Case Study 1: xAODAnaHelpers

- xAODAnaHelpers is a general analysis framework
- Documentation is hosted in two places:
  - ReadTheDocs: <https://xaodanahelpers.readthedocs.io/en/ma>
  - GitHub Pages: <https://ucatlas.github.io/xAODAnaHelpers/>
- Documentation is written in two places
  - C++ doxygen comments in header files
  - Python docstrings in python scripts/packages
- Documentation is organized using RST (reStructured Text) files
- Glued together using Doxygen+Breathe+Sphinx with continuous integration
  - ReadTheDocs has a webhook to trigger builds, built from continuous integration

## xAODAnaHelpers

DOI [10.5281/zenodo.998269](https://doi.org/10.5281/zenodo.998269) build passing

AnalysisBase,RootCore 2.4.37 AnalysisBase,CMake 21.2.10

The xAOD analysis framework, born out of ProofAna...or not.

Welcome to the xAODAnaHelpers wiki! This is an xAOD Analy ATLAS.

## Doxygen Comments

```
/**
 * @rst
 *
 * This is used by all algorithms within |xAH|.
 *
 * .. note:: The expectation is that the user does
 *
 * The main goal of this algorithm class is to sta
 *
 * We expect the user to create a new algorithm, s
 *
 * class JetSelector : public xAH::Algorithm
 * {
 * // ...
 * };
```

This is used by all algorithms within xAODAnaHelpers.

### Note

The expectation is that the user does not directly use this class but rath

The main goal of this algorithm class is to standardize how everyone defri plugs into xAODAnaHelpers. A series of common utilities are provided su which defines the class name so we can manage a registry `m_instanceReg` xAODAnaHelpers as flexible as possible to our users.

We expect the user to create a new algorithm, such as a selector for jets:

```
class JetSelector : public xAH::Algorithm
{
// ...
};
```

## Output

# Case Study 2: Ironman

- L1Calo communication software for embedded OS
- Documentation is hosted in ReadTheDocs
  - ReadTheDocs: <https://xaodanahelpers.readthedocs.io/en/master/>
- Documentation is written in Python
  - Allows for automatic doctesting to ensure your documentation does not get stale and represents how your code should function

## Building an IPBus Packet

Because of duck-typing, any object can make do with original packet which has a packet id `:code:`0x0``

```
>>> from ironman.constructs.ipbus import IPBusConstruct
>>> data = '\x20\x00\x00\xf0\x20\x00\x01\x0f\x00'
>>> p = IPBusConstruct.parse(data)
>>> p.header.packet_id = 0x1
>>> new_data = IPBusConstruct.build(p)
>>> print repr(new_data)
'\x00\x00\xf0 \x00\x01\x0f\x00\x00\x00\x03'
>>>
```

## Building an IPBus Packet

Because of duck-typing, any object can make do when *construct* docs for more information here. In this case packet id `0x0` in the header and update it to `0x1`

```
>>> from ironman.constructs.ipbus import IPBusConstruct
>>> data = '\x20\x00\x00\xf0\x20\x00\x01\x0f\x00'
>>> p = IPBusConstruct.parse(data)
>>> p.header.packet_id = 0x1
>>> new_data = IPBusConstruct.build(p)
>>> print repr(new_data)
'\x00\x00\xf0 \x00\x01\x0f\x00\x00\x00\x03'
>>>
```

## Ironman

python package 0.2.15 docs latest

build passing codecov 85% health 97%

### What is Ironman?

Ironman is a general purpose software toolbox to be run on L1Calo hardware with embedded processors (SoCs).

Look how easy it is to use

```
>>> import ironman
>>> # Get your stuff done
>>> ironman.engage()
```

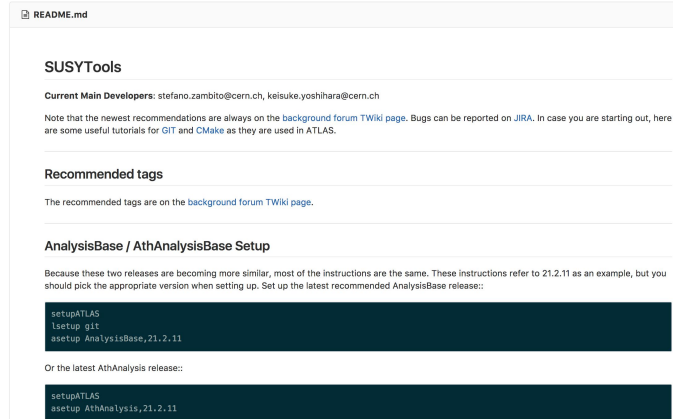
### Features

- Be awesome
- Make things faster



# Case Study 3: SUSYTools

- SUSYTools is part of the ASG software releases
- Documentation in two places
  - For developers: [README.md](#) on [GitLab](#)
  - For physicists/analyzers: on a [twiki](#)
- This is nice as both the twiki and GitLab README cross-link each other
  - If you know how to find one set of docs, you can find the other set of docs
- Since each documentation has an audience focus, splitting it up makes sense
- README is in Markdown format, plain-text, easy to format
  - Generally follows release more tightly as it is part of git commits
  - Twiki is not tied to commits in [atlas/athena](#), **physics recommendations often follow software releases**





# Case Study 4: Jet/MET

- Jet/MET is a CP group in ATLAS
- A tutorial in release 21 is on GitLab
  - [https://gitlab.cern.ch/atlas-jetmiss/JetRecoTutorial\\_R21](https://gitlab.cern.ch/atlas-jetmiss/JetRecoTutorial_R21)
- Shows how to run jet finding, grooming, and substructure
  - Using C++ and calls to fastjet
  - Using ATLAS JetRec package
  - Using Python entirely to configure everything
- Very nice as example code and tutorial are together
  - Keeping everything in one “module” makes it easier for a new user
- Entire tutorial is a single markdown file (README.md)
  - Tutorial is located under atlas-jetmiss, maybe a central location for tutorials would be better? [all CP group tutorials in one place?]

## Table of Contents

---

Description

Quick Setup

Tutorial 1 : Native fastjet in ATLAS

Tutorial 2 : JetRec in ATLAS

Tutorial 3 : Advanced JetRec Tools via jobOptions

Tutorials Explained

Run an Example on the Grid

ToDo List

## Authors :

---

These are the main authors and initial developers of this. If you *should* have questions, and you should *ask* these people for

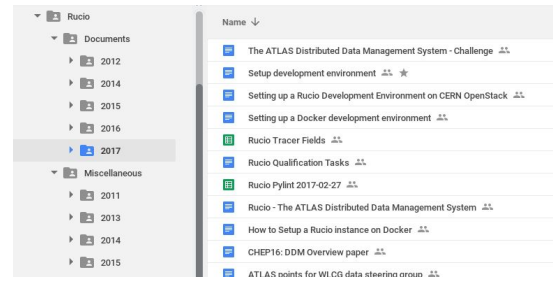
- Sam Meehan [samuel.meehan@cern.ch](mailto:samuel.meehan@cern.ch)
- Pierre-Antoine Delsart [delsart@in2p3.fr](mailto:delsart@in2p3.fr)

# Case Study 5: ASG Software Docs

- As part of migration to release 21, large effort to write new docs
- Documentation is hosted on [CERN DFS](#) and maintained in GitLab
  - <https://gitlab.cern.ch/atlas-sw-git/atlassoftwaredocs/>
- A nice example of using CERN-only applications/packages for documentation
  - GitLab CI builds the documentation and deploys it
  - Jekyll (a blogging platform) is used to generate the static site (uses Ruby and markdown files)
- Is DFS a good fit for this? (see [knowledgebase article](#) for EOS alternative)
  - Note that CERN disabled GitLab pages in favor of DFS and EOS because of a lack of SSO support (reasonable)
  - [DFS requires an active user to own](#), but how to persist after that person leaves? What if we forget?

# Case Study 6: Rucio and DDM

- Three types of documentation (Tech, Ops, Public)
- Technical: Code and API
  - Started with PyDoc but now moved to Sphinx (easier)
- Operational: Clients
  - The same, however common complaints about old CLI examples
- Operational: Howtos
  - Basically ATLAS operational documentation, everything in TWikis
- Operational: Meetings, Reports, Incidents
  - Collaborative editing in Google Docs – Diagramming must have!
  - Weekly meetings are fully in Indico minutes
- Public: Presentations and Online Support
  - Shared Google Slides folder with a single template
  - Papers: Collaborative LaTeX editing using Overleaf
  - Online support: Slack (also for outside ATLAS support)

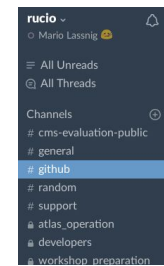


```
106 @transactional_session
107 def add_did(scope, name, type, account, statuses=None, meta=None, rules=None,
108           lifetime=None, dds=None, rse=None, session=None):
109     ...
110     Add data identifier.
111
112     :param scope: The scope name.
113     :param name: The data identifier name.
114     :param type: The data identifier type.
115     :param account: The account owner.
116     :param statuses: Dictionary with statuses, e.g.g {'monotonic': True}.
117     :meta: Meta-data associated with the data identifier is represented using key/value pairs in a
118     :rules: Replication rules associated with the data identifier. A list of dictionaries, e.g., [
119     :param lifetime: DID's lifetime (in seconds).
120     :param dds: The content.
121     :param rse: The RSE name when registering replicas.
122     :param session: The database session in use.
123     ...
124     return add_did(dids=[{'scope': scope, 'name': name, 'type': type,
```

```
Table of Contents
Installing ATLAS Rucio
Clients
Prerequisites
Python Dependencies
Setup CERN AFS client
on lxplus
Install via pip
Upgrade via pip

Rucio clients need the following python modules:

argparse>=1.4.0 # Command-line parsing library
argcomplete>=1.9.2 # Bash tab completion for argparse
requests>=2.18.4 # Python HTTP for Humans.
urllib3>=1.22 # HTTP library with thread-safe co
dogpile.cache>=0.6.4 # Caching API plugins
nose>=1.3.7 # For rucio test-server
# S2 boto protocol
nt>=3.4.0 # OpenStack Object Storage API Cl
34.2 # Pretty-print tabular data
4.13 # Test progress bar
# Read and write bzip2-compressed
# File type identification using l
# Clean single-source support for
# Python 2 and 3 compatibility uti
```



### #github

2017-12-05

1. Release schedule
  - a. [1.14.0 post1](#) on PROD today
  - b. [1.14.1 post1](#) goes on INT today
    - i. Goes on ALRB testing today
    - ii. Default next week
  - c. 1.14.2 next week
2. Popularity for datasets accessed outside grid
  - a. Will present results of local access to next CREM

# Moving forward

# Summary

- Main challenges
  - **Maintenance and education**
  - Automate the creation of documentation from the code
  - Crosslink documentation between projects
  - Diagramming is a pain
  - Collaborative editing of "operational documentation"
  - Removing outdated documentation / duplicates
  - **Making documentation searchable / discoverable**
- The tools are there, it doesn't look like a technological problem
  - (never mind that twiki is slow as a snail and hates concurrent updates)
- Proposed next steps
  - **??? DISCUSS!**