



Doxygen for DUNE DAQ

Andrew J. Mogan

DUNE DAQ Software Coordination Meeting

June 13, 2024

Doxygen

- Automated documentation generator
- Parses many file types, notably C++ and Python source files
- Gathers information on classes, namespaces, etc.
 - Dependency graphs
 - Cross-referencing (automatically generates links)
- Motivation: we now have the potential for many generated classes in the v5/develop line
 - OKS classes spun off of xml files
- We now have a nightly Doxygen workflow to keep up-to-date documentation of these classes etc.
 - Currently only for v5/develop line
- The following slides cover how this works; the result is published to the [DUNE DAQ Github Pages](#)

The screenshot shows the Doxygen web interface. On the left is a navigation pane with a tree view of the project structure, including 'Cache'. The main content area displays the documentation for the `find()` method. At the top right of the content area, there is a link: 'Example from [Doxygen homepage](#)'. The documentation includes the method signature: `template<typename K, typename V> V * Cache<K, V>::find (const K & key)`. Below the signature, it states: 'Finds a value in the cache given the corresponding key.' The 'Returns' section says: 'a pointer to the value or nullptr if the key is not found in the cache'. A yellow 'Note' box contains the text: 'The hit and miss counters are updated, see hits() and misses()'. The 'Definition at line 105 of file cache.h.' section shows the C++ code for the method. At the bottom of the interface, there is a 'Show dark mode output' toggle and a footer that says 'Generated by doxygen 1.10.0'.

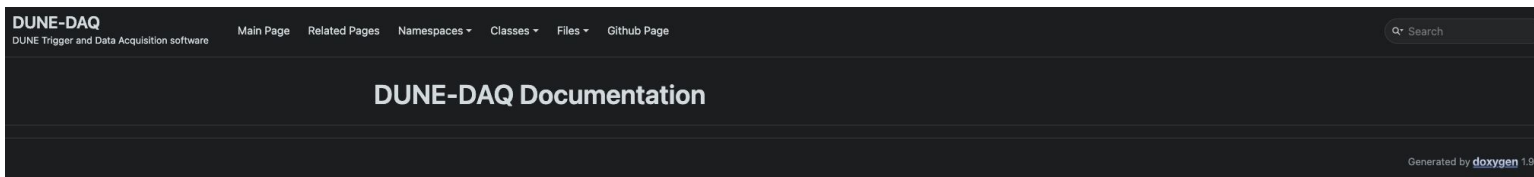
```
106 {
107     auto it = m_cacheItemMap.find(key);
108     if (it != m_cacheItemMap.end())
109     {
110         // move the item to the front of the list
111         m_cacheItemList.splice(m_cacheItemList.begin(),
112                               m_cacheItemList,
113                               it->second);
114         m_hits++;
115         // return the value
116         return &it->second->second;
117     }
118     else
119     {
120         m_misses++;
121     }
122     return nullptr;
123 }
```

How Doxygen Works (in General)

- In general, Doxygen is straightforward to [install and run](#)

```
INSTALL_DIR=/home/$USER/doxygen-install-dir # Or whatever
wget https://www.doxygen.nl/files/doxygen-1.10.0.src.tar.gz
gunzip doxygen-1.10.0.src.tar.gz
tar xf doxygen-1.10.0.src.tar
cd doxygen-1.10.0/
mkdir build
cd build
cmake -G "Unix Makefiles" \
      -DCMAKE_INSTALL_PREFIX=$INSTALL_DIR .. # Else permission complaints
make && make install
export PATH=$INSTALL_DIR/bin:$PATH
cd /path/to/project
doxygen -g # Generates Doxyfile; modify accordingly, then run
doxygen
```

- Should get an index.html file as output; point a web server to this file



How Doxygen Works – Documenting the Code

- Doxygen parses header and source files for special comment blocks

```
1  /**
2   * @file DataStore.hpp
3   *
4   * This is the interface for storing and retrieving data from
5   * various storage systems.
6   *
7   * This is part of the DUNE DAQ Application Framework, copyright 2020.
8   * Licensing/copyright details are in the COPYING file that you should have
9   * received with this code.
10  */
11
12  // 09-Sep-2020, KAB: the initial version of this class was based on the
13  // Queue interface from tl
14
```

Detailed Description

This is the interface for storing and retrieving data from various storage systems.

This is part of the DUNE DAQ Application Framework, copyright 2020. Licensing/copyright details are in the COPYING file that you should have received with this code.

Definition in file [DataStore.hpp](#).

How Doxygen Works – Documenting the Code

- Doxygen parses header and source files for special comment blocks

```
/**  
 * @brief Informs the DataStore that writes or reads of data blocks associated  
 * with the specified run number have finished, for now.  
 * This allows DataStore instances to do any cleanup or shutdown operations  
 * that are useful once the writes or reads for a given run number have finished.  
 */  
virtual void finish_with_run(daqdataformats::run_number_t run_number) = 0;
```

Member Function Documentation

◆ finish_with_run()

virtual void dunedaq::dfmodules::DataStore::finish_with_run (daqdataformats::run_number_t run_number)

pure virtual

Informs the [DataStore](#) that writes or reads of data blocks associated with the specified run number have finished, for now. This allows [DataStore](#) instances to do any cleanup or shutdown operations that are useful once the writes or reads for a given run number have finished.

◆ operator=() [1/2]

[DataStore](#) & dunedaq::dfmodules::DataStore::operator= (const [DataStore](#) &)

private

delete

The Workflow










- The workflow lives in the [docs repo](#) →
- Automatically triggered nightly at 5 a.m. UTC
 - Two hours after the nightly A9 build starts so as to have access to said build
- Runs on a GitHub-hosted runner in the container output by the nightly build (has cvmfs access)
- Alessandro made some handy scripts while testing Doxygen; those are now in [daq-release](#)
 - Starts with template Doxyfile and overwrites the input directories based on what's available
 - triggeralgs is currently excluded for technical reasons
- Creates a local dev area, checks out all fddaq and coredaq packages into it, and runs Doxygen
- Publishes result to [GitHub Pages](#)
- Takes less than 20 minutes total

```
name: Build Doxygen and publish to GitHub pages

on:
  schedule:
    - cron: "0 5 * * *"

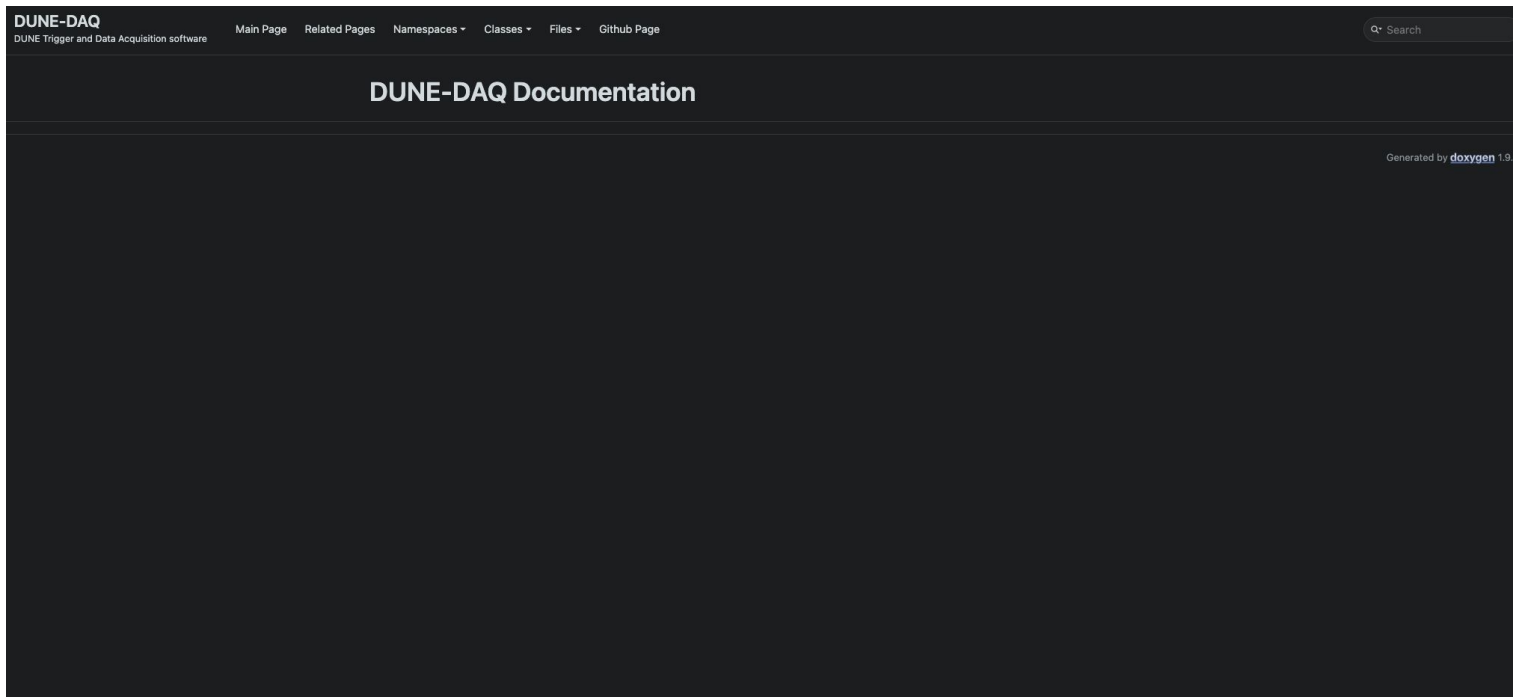
jobs:
  build-doxygen:
    name: build_doxygen
    runs-on: ubuntu-latest
    container:
      image: "ghcr.io/dune-daq/nightly-release-alma9:development_v5"
    env: --
    defaults: --

    steps:
      - name: Checkout daq-release --
      - name: Create dbt area --
      - name: Checkout packages --
      - name: Build dbt area --
      - name: Generate Doxyfile --
      - name: Doxygen Action --
      - name: Deploy --
```

 Build Doxygen and publish to GitHub pages	develop	 14 hours ago  17m 54s
 Build Doxygen and publish to GitHub pages	develop	 2 days ago  18m 20s
 Build Doxygen and publish to GitHub pages	develop	 3 days ago  17m 55s

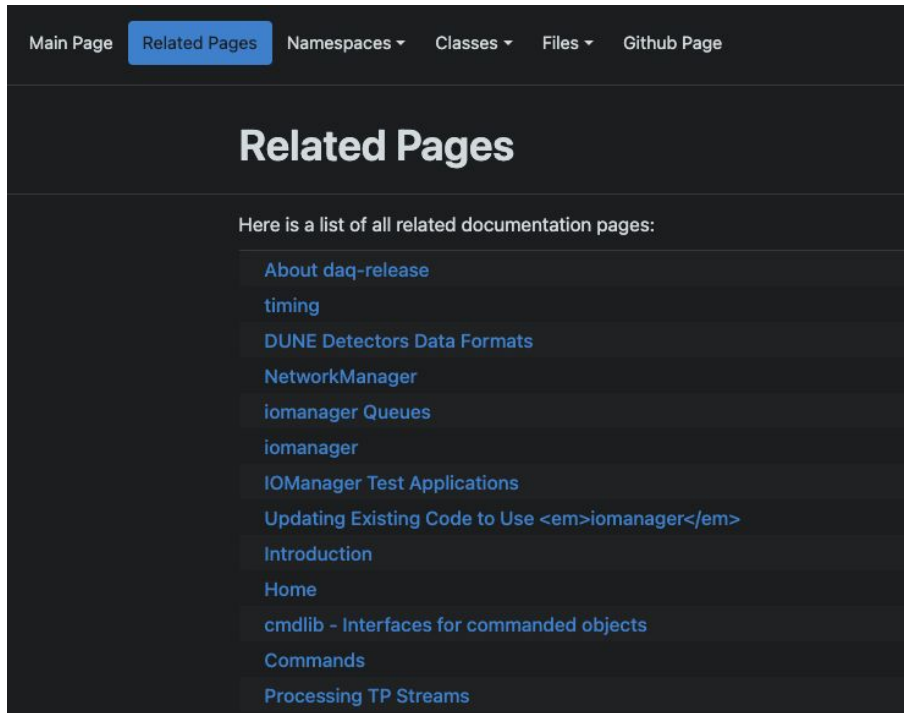
The Result – Homepage

- It works!
 - ...but could use some polishing



Related Pages

- Contains various links to READMEs and other documentation pages



The screenshot shows a dark-themed navigation bar with the following items: 'Main Page', 'Related Pages' (highlighted in blue), 'Namespaces ▾', 'Classes ▾', 'Files ▾', and 'Github Page'. Below the navigation bar, the heading 'Related Pages' is displayed in large white text. Underneath, a subtitle reads 'Here is a list of all related documentation pages:'. A list of links follows, including 'About daq-release', 'timing', 'DUNE Detectors Data Formats', 'NetworkManager', 'iomanager Queues', 'iomanager', 'IOManager Test Applications', 'Updating Existing Code to Use `iomanager`', 'Introduction', 'Home', 'cmdlib - Interfaces for commanded objects', 'Commands', and 'Processing TP Streams'.

Related Pages

- Contains various links to READMEs and other documentation pages

Main Page **Related Pages** Namespaces ▾ Classes ▾ Files ▾ Github Page

Related Pages

Here is a list of all related documentation pages:

- About daq-release**
- timing
- DUNE Detectors Data Formats
- NetworkManager
- iomanager Queues
- iomanager
- IOManager Test Applications
- Updating Existing Code to Use `iomanager`
- Introduction
- Home
- cmdlib - Interfaces for command objects
- Commands
- Processing TP Streams

Related Pages Namespaces ▾ Classes ▾ Files ▾ Github Page

About daq-release

- Alma9 build v4 production nightly release **passing**
- SL7 build v4 production nightly release **passing**
- Alma9 build v5 nightly release **passing**
- SL7 build v5 nightly release **passing**
- Nightly integration tests **passing**

This is a repo containing DUNE DAQ release making tools, configuration files, and build scripts for both DUNE-DAQ and external packages.

Table of contents

For DAQ software developers and users:

- DAQ software development workflow
- List of GitHub Teams and Repositories
- How to build external software with Spack in a local workarea

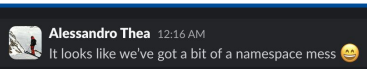
For the Software Coordination team (expert only):

- Nightly Releases and Continuous Integration
- Creating a new DAQ release
- How to publish files to cvmfs
- How to build a new stack of external software

Single-Repo CI Build Status

- appmodel: build-develop **passing**
- appfwk: build-develop **passing**
- cmdlib: build-develop **passing**

The Result – Namespaces



DUNE-DAQ
DUNE Trigger and Data Acquisition software

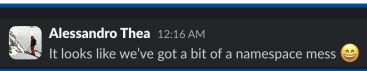
Main Page Related Pages Namespaces ▾ Classes ▾ Files ▾ Github Page

Namespace List

Here is a list of all namespaces with brief descriptions:

- Namespace `__main__`
- Namespace `afc`
 - Namespace `click_texttable`
 - Class `ArraySizeError`
 - Class `bcolors`
 - Class `Texttable`
 - Namespace `crossbar`
 - Namespace `ipmi`
 - Namespace `mmc`
 - Namespace `toolbox`
 - Class `IntRange`
- Namespace `appfwk`
 - Namespace `utils`
 - Namespace `appmodel`
 - Namespace `assets`
 - Namespace `binary_dump`
- Namespace `boost`
 - Namespace `interprocess`
 - Class `shared_ptr`
 - Namespace `cardcontrollerapp_gen`
 - Namespace `checkLinks`
- Namespace `conffwk`
 - Namespace `ConfigObject`
 - Class `_ConfigObjectProxy`
 - Class `ConfigObject`
 - Namespace `Configuration`

The Result – Namespaces



DUNE-DAQ
DUNE Trigger and Data Acquisition software

Main Page Related Pages Namespaces ▾ Classes ▾ Files ▾ Github Page

Namespace List

Here is a list of all namespaces with brief descriptions:

- ▾ [N](#) __main__
- ▾ [N](#) afc
 - ▾ [N](#) click_texttable
 - [C](#) ArraySizeError
 - [C](#) bcolors
 - [C](#) Texttable
 - [N](#) crossbar
 - [N](#) ipmi
 - [N](#) mmc
 - ▾ [N](#) toolbox
 - [C](#) IntRange
 - ▾ [N](#) appfwk
 - [N](#) utils
 - [N](#) appmodel
 - [N](#) assets
 - [N](#) binary_dump
 - ▾ [N](#) boost
 - [N](#) interprocess
 - [C](#) shared_ptr
 - [N](#) cardcontrollerapp_gen
 - [N](#) checkLinks
 - ▾ [N](#) conffwk
 - [N](#) ConfigObject
 - [C](#) _ConfigObjectProxy
 - [C](#) ConfigObject
 - ▾ [N](#) Configuration

appfwk.utils Namespace Reference

Functions

- [mspec](#) (inst, plugin, conn_refs)
- [acmd](#) (list mods)
- [mcmd](#) (str cmdid, list mods)
- [mrccmd](#) (cmdid, instate, outstate, mods)

Function Documentation

◆ [acmd\(\)](#)

`appfwk.utils.acmd(list mods)`

Helper function to create appfwk's CmdObj of Addressed module commands.

:param cmdid: The coomand id
:type cmdid: str
:param mods: List of module name/data structures
:type mods: list

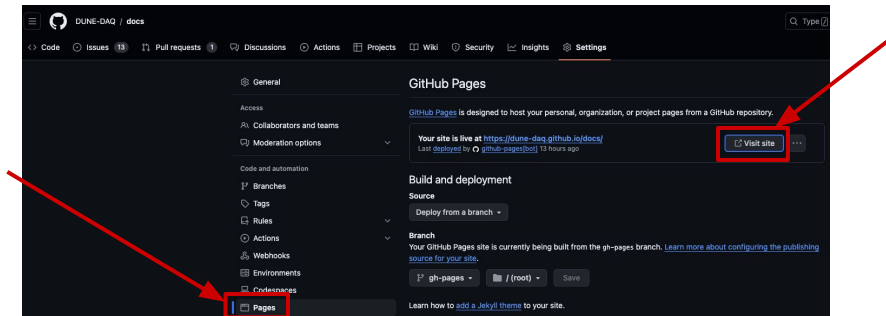
:returns: A constructed Command object
:rtype: dunedaq.appfwk.cmd.Command

Definition at line 35 of file `utils.py`.

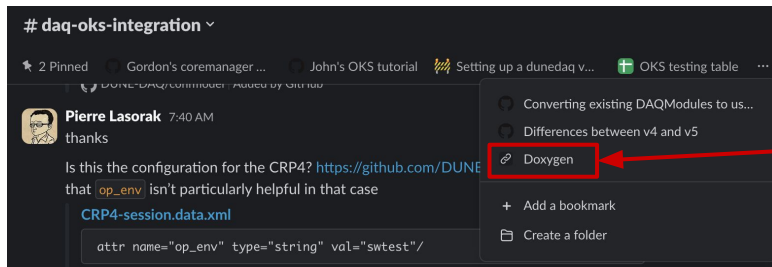
```
35 def acmd(mods: list):
36     """
37     Helper function to create appfwk's CmdObj of Addressed module commands.
38
39     :param cmdid: The coomand id
40     :type cmdid: str
41     :param mods: list of module name/data structures
```

Getting to the Doxygen Page

- Option 1: The Hard Way
 - Go to [docs](#) repo in the DUNE DAQ organization GitHub, Settings -> Pages -> Visit Site



- Option 2: Bookmark in the #daq-oks-integration
 - May need to click the three dots to see it



Click the three dots

Summary

- The DUNE DAQ Doxygen page is now live on [GitHub Pages](#) and updated nightly
- Discussion points:
 - The page style and layout could be adjusted if people want
 - Where else should we put the link for maximum visibility?
 - Other thoughts?