



STScI | SPACE TELESCOPE
SCIENCE INSTITUTE

EXPANDING THE FRONTIERS OF SPACE ASTRONOMY

The JWST Quicklook Application (JWQL)



Matthew Bourque
(Technical Lead)



Francesca Boffi
(Project Manager)



Lauren Chambers
(FGS)



Misty Cracraft
(MIRI)



Joseph Filippazzo
(NIRISS)



Bryan Hilbert
(NIRCam)



Graham Kanarek
(NIRSpec)



Catherine Martlin
(WFC3)



Sara Ogaz
(DATB)



Johannes Sahlmann
(NIRISS)



Acknowledgements

INS

Tracy Beck
Rosa Diaz
Van Dixon
Scott Friedman
Alex Fullerton
Catherine Goscmyer
Dean Hines
Sherrie Holfetz
Margaret Meixner
Massimo Robberto
Linda Smith

DMD

Faith Abney
Anastasia Alexov
Sara Anderson
Clara Brasseur
Matthew Burger
Steven Crawford
James Davies
Tom Donaldson
Lisa Gardner

Joe Hunkeler
Catherine Kaleida
Mark Kyprianou
Karen Levay
Lee Quick
Matt Rendina
Mary Romelfanger
Bernie Shiao
Geoff Wallace

SCOPE
Crystal Mannfolk

SEITO
Maria Nieto-Santisteban

ITSD

Vera Gibbs
Phil Grant
Greg Masci
Prem Mishra
Don Meuller
Anupinder Rai
Matthew Sienkiewicz
Patrick Taylor
Dave Unger
Thomas Walker
Alex Yermolaev
Joe Zahn

OPO

Lara Wilkinson
DSMO
Michael Fox
Afon Smith
JWSTMO
Joe Pollizzi
Jeff Valenti



Goals & Motivation

Visually inspect new and archived data

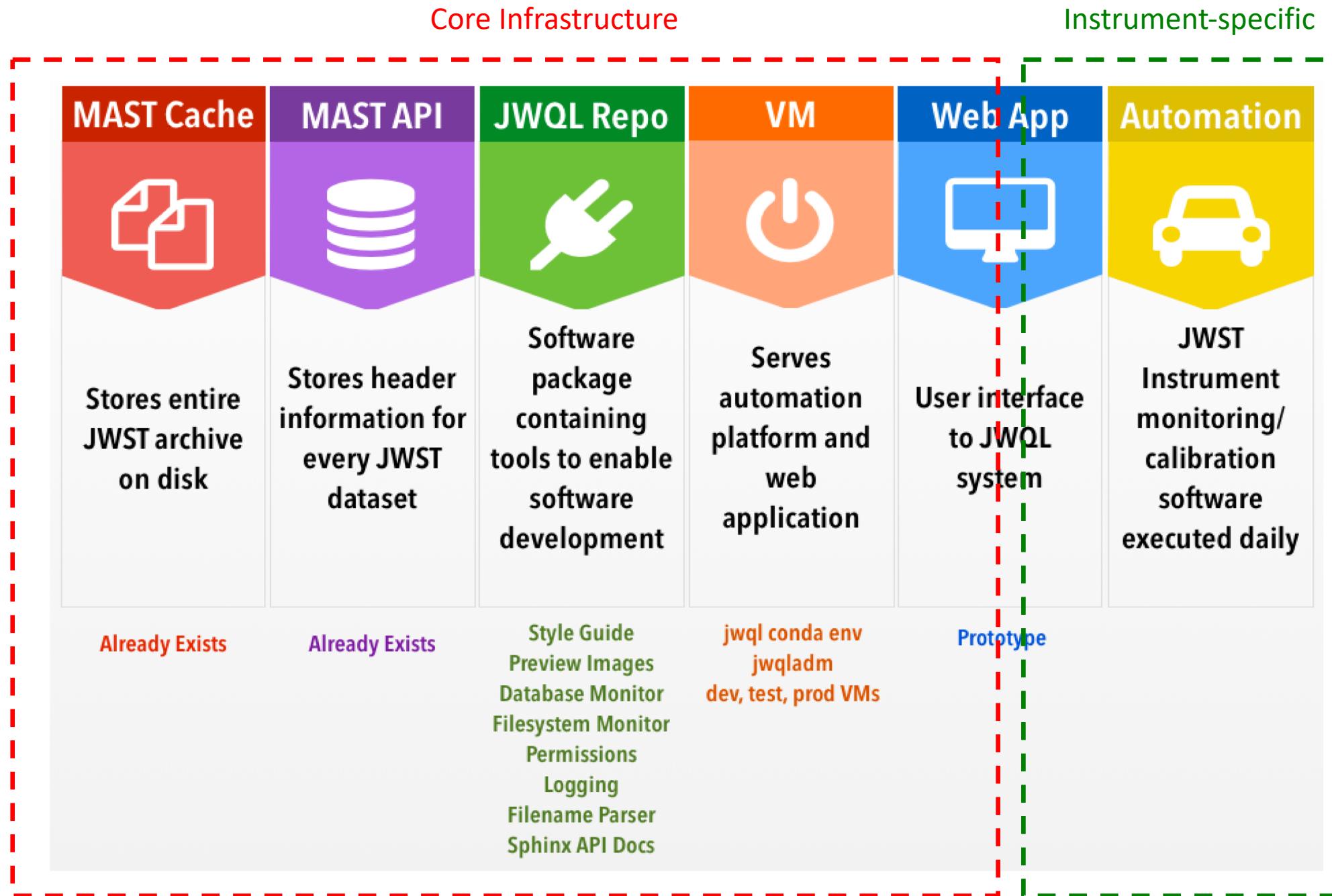
Automated calibration & monitoring software

Convenience of a web browser

One solution for five teams

A database-driven web application and automation framework for analyzing and monitoring JWST instruments and data

MAST Cache	MAST API	JWQL Repo	VM	Web App	Automation
					
Stores entire JWST archive on disk	Stores header information for every JWST dataset	Software package containing tools to enable software development	Serves automation platform and web application	User interface to JWQL system	JWST Instrument monitoring/calibration software executed daily
Already Exists	Already Exists	Style Guide Preview Images Database Monitor Filesystem Monitor Permissions Logging Filename Parser Sphinx API Docs	jwql conda env jwqladm dev, test, prod VMs	Prototype	



MAST Cache



Stores entire
JWST archive
on disk

```
(astroconda3) [dljwql:jw00318001001] ls *.fits
jw00318001001_02101_00001_nis_cal.fits
jw00318001001_02101_00001_nis_i2d.fits
jw00318001001_02101_00001_nis_original.fits
jw00318001001_02101_00001_nis_rate.fits
jw00318001001_02101_00001_nis_rateints.fits
jw00318001001_02101_00001_nis_trapsfilled.fits
jw00318001001_02101_00001_nis_uncal.fits
jw00318001001_02102_00001_nis_cal.fits
jw00318001001_02102_00001_nis_i2d.fits
jw00318001001_02102_00001_nis_original.fits
jw00318001001_02102_00001_nis_rate.fits
jw00318001001_02102_00001_nis_rateints.fits
jw00318001001_02102_00001_nis_trapsfilled.fits
jw00318001001_02102_00001_nis_uncal.fits
```



Stores header information for every JWST dataset

```
from astroquery.mast import Mast

service = "Mast.Jwst.Filtered.Nircam"
params = {"columns": "filename, instrume, detector, filter,\n            expstart, apername, duration",
          "filters": [{"paramName": "filename",
                       "values": ['jw94015001002_02104_00001_nrca2_cal.fits']}
                      ]}
response = Mast.service_request_async(service,params)
result = response[0].json()['data']

result

[{'apername': 'NRCA2_FULL',
 'detector': 'NRCA2',
 'duration': 64.42056000000001,
 'expstart': 57410.63391019028,
 'filename': 'jw94015001002_02104_00001_nrca2_cal.fits',
 'filter': 'F115W',
 'instrume': 'NIRCAM'}]
```



Software
package
containing
tools to enable
software
development

We use a standardized, open development workflow using git/GitHub that:

- **Enables Collaboration** – Many people contributing to the code base simultaneously
- **Supports Continuous Delivery** – “Production” version of code that is well tested and works
- **Aids in code readability & maintainability**
- **Avoids redundancy** – One solution instead of five
- **Facilitates knowledge transfer** – At least two people looking at every line of code
- **Promotes industry and institute best practices**
- **Conforms to institute security policies**

This is accomplished by:

- Cross-instrument / cross-division development team
- Development standards
- Code Review
- Documentation for onboarding and contributing



**Software
package
containing
tools to enable
software
development**

Continuous Integration

Version Control Policy

GitHub Policy

Source Code Control

Created by Glenn Miller, last modified on Apr 05, 2018

Purpose	<p>Source code control is a best practice for software engineering. Source code control has evolved to embrace the iterative and collaborative models of code development seen across the industry. The products and services we provide to our customer, NASA, are embodied in the software and other products that we develop. Rights to these products are retained by NASA, and all of the Intellectual Property rights are retained by AURA. Therefore, we have an obligation to retain our software and other products in perpetuity. Benefits of this policy are:</p> <ul style="list-style-type: none">• Cross-Institute support for best practices• Decreased cost of infrastructure support• Ability to meet contractual requirements to deliver, store, track changes to, and archive our products <p>Please direct questions about this policy to the IT Service Desk, itsupport.stsci.edu. Read more about enforcement or exceptions to policy.</p>
---------	---

GitHub

Created by Glenn Miller, last modified on Feb 23, 2018

Purpose	<p>Producing software (in the form of source code and compiled binaries), is a core part of what we do at STScI. Sometimes this software needs to be shared with collaborators and the wider community. This policy is designed to govern how we make use of the GitHub platform for collaborating on public and private repositories.</p> <p>Please direct questions about this policy to the IT Service Desk, itsupport.stsci.edu. Read more about enforcement or exceptions to policy.</p>
---------	--

Continuous Integration (Draft)

Created by Arfon Smith, last modified by Glenn Miller on Feb 27, 2018

Purpose	<p>Continuous Integration (CI), is a development practice whereby developers check-in code to a shared repository regularly (usually several times per day). Each time this code is checked-in, the automated test suite is executed, thereby allowing teams to detect problems with their codebase early.</p> <p>Continuous Integration, combined with automated test suites and Source Code Control, are the foundations of modern best practice Software Engineering. Our customer, NASA, requires us to produce reliable, high-quality software, delivered on a regular schedule. Implementing this policy can only serve to help us deliver on this core competency of the Institute.</p> <p>Please direct questions about this policy to the IT Service Desk, itsupport.stsci.edu. Read more about enforcement or exceptions to policy.</p>
---------	---

JWQL Repo



Software
package
containing
tools to enable
software
development

github.com/spacetelescope/jwql

spacetelescope/jwql: The James Webb Quicklook Application

Code Issues Pull requests Projects Wiki Insights Settings

425 commits 3 branches 31 releases 11 contributors BSD-3-Clause

Branch: master New pull request Create new file Upload files Find file Clone or download

bourque Merge pull request #129 from laurenmarietta/move-website-to-package ... Latest commit f0e719c 21 hours ago

docs Merge branch 'master' into anomoly_db 6 days ago

jwql Move website directory into jwql package; a couple bug fixes along th... 2 days ago

notebooks Added colnames property to Anomaly class and updated notebook. 23 days ago

style_guide Added link to example.py 3 months ago

typing_demo Adding type annotations demo files 8 months ago

.gitignore Merge branch 'master' of github.com:spacetelescope/jwql into web-app-dev 2 months ago

CODE_OF_CONDUCT.md Initial commit of code of conduct 2 months ago

Jenkinsfile Unsurpressing output for pytest 2 months ago

LICENSE Added LICENCSE and .gitignore. Initial commit of repository. 10 months ago

MANIFEST.in Added a few more attributes to the setup script. Initial commit of MA... 6 months ago

README.md Small stylistic updates; change home page layout & description 14 days ago

environment.yml Merge branch 'master' into anomoly_db 6 days ago

setup.py Forgot that two equal signs are required : 2 months ago

README.md

JWQL

The James Webb Quicklook Application (JWQL)

JWQL Repo



Software
package
containing
tools to enable
software
development

- PEP8 Python Standards
- PEP257 Documentation Standards
- numpydoc API doc conventions

jwql/style_guide.md at master · GitHub

spacetelescope / jwql · Private

Code Issues 14 Pull requests 4 Projects 0 Wiki Insights Settings

Branch: master · jwql / style_guide / style_guide.md

bourque Added comment about OWASP top 10 58d497d on Feb 21

2 contributors

75 lines (44 sloc) | 3.79 KB Raw Blame History

Python Code Style Guide for jwql

This document serves as a style guide for all `jwql` software development. Any requested contribution to the `jwql` code repository should be checked against this guide, and any violation of the guide should be fixed before the code is committed to the `master` branch. Please refer to the accompanying `example.py` script for a example code that abides by this style guide.

Prerequisite Reading

It is assumed that the reader of this style guide has read and is familiar with the following:

- The [PEP8 Style Guide for Python Code](#)
- The [PEP257 Docstring Conventions Style Guide](#)
- The [numpydoc docstring convention](#)

Workflow

All software development for the `jwql` project should follow a continuous integration workflow, described in the [git & GitHub workflow for contributing](#). Before committing any code changes, use `flake8` to check the code against `PEP8` standards. Also check that your code is conforming to this style guide.

Version Numbers and Tags

Any changes pushed to the `master` branch should be tagged with a version number. The version number convention is `x.y.z`, where

`x` = The main version number. Increase when making incompatible API changes.
`y` = The feature number. Increase when change contains a new feature with or without bug fixes.
`z` = The hotfix number. Increase when change only contains bug fixes.

JWQL Repo



Software package containing tools to enable software development

All code is reviewed through Pull Requests

Filename parser utility function #63

Merged bourque merged 6 commits into `spacetelescope:master` from `laurenmarietta:filename_parser` on Apr 19

Conversation 15 Commits 6 Checks 0 Files changed 2 +138 -2

laurenmarietta commented on Apr 16 Closes #61

laurenmarietta added some commits on Apr 16

- Add Joe Hunkeler's filename parser to `utils.py` with minor modifications ef7c148
- Update references/authors f55f25f

laurenmarietta added enhancement, Code Base, Low Priority labels on Apr 16

laurenmarietta requested a review from bourque on Apr 16

laurenmarietta commented on Apr 16 @bourque if you want to assign someone else other than yourself to review, please feel free!

bourque requested a review from SaOgaz on Apr 16

bourque commented on Apr 16 @saraogaz will review this one

bourque assigned laurenmarietta on Apr 16

SaOgaz reviewed on Apr 16 + edited View changes

Hey @laurenmarietta, this looks good to me, but would you mind adding a pytest for this function? Shouldn't be too bad to do, let me know if you want some help with it.

Reviewers: bourque (✓), SaOgaz

Assignees: laurenmarietta

Labels: Code Base (yellow), Low Priority (green), enhancement (light green)

Projects: None yet

Milestone: No milestone

Notifications: Unsubscribe

You're receiving notifications because you modified the open/close state.

3 participants: laurenmarietta, bourque, SaOgaz

Lock conversation

JWQL Repo



Software
package
containing
tools to enable
software
development

git & GitHub workflow for contributing

Matthew Bourque edited this page on Mar 13 · 3 revisions

The best method for contributing software to the `jwql` project is a workflow that involves forking the `jwql` repository, developing changes on personal branches, and opening pull requests through GitHub.

The first question you will have to figure out is whether you should open an issue for this update - if you think that this change will be solving a significant problem or add a significant enhancement to the project then it would be advantageous to open an issue ticket [here](#). This will allow both individuals and the team as a whole to keep track of the project and our progress as we go. Any appropriate individuals should be assigned to the issue, and a label(s) should be tagged.

Following that, any changes that you want to eventually make to the `master` branch should be done through the workflow where you create a fork and work on your own branch before submitting those changes to be reviewed through a pull request. Instructions on how to do those things can be found below:

1. Create a personal fork of the `jwql` repository by visiting the [spacetelescope repository](#) and clicking the `Fork` button. Note that this only has to be done once.
2. Make a local copy of your personal fork by cloning the repository (e.g. `git clone https://github.com/username/jwql.git`, found by clicking the green "clone or download" button.). Note that, unless you explicitly delete your clone of the fork, this only has to be done once.
3. Ensure that the personal fork is pointing to the upstream `spacetelescope/jwql` repository with `git remote add upstream https://github.com/spacetelescope/jwql.git`. Note that this only has to be done once.
4. Create a branch on the cloned personal fork to develop software changes on. Branch names should be short but descriptive (e.g. `new-database-table` or `fix-ingest-algorithm`) and not too generic (e.g. `bug-fix`). Also consistent use of hyphens is encouraged.
 - i. `git branch <branchname>` - you only need to do this when you first create your branch.
 - ii. `git checkout <branchname>` - you can use this command to switch back and forth between existing branches.
 - iii. Perform local software changes using the nominal `git add / git commit -m` cycle.
 - a. `git status` - allows you to see which files have changed.

Pages 8

- Home
- Config file
- Environment Installation
- Feature Requests
- git & GitHub workflow for contributing
- JWQL Logging Function: The Why's and The How
- Useful Links
- Web Application

+ Add a custom sidebar

Clone this wiki locally

<https://github.com/spacetelescope/jwql>

Clone in Desktop



Software package containing tools to enable software development

Core Infrastructure Development Progress

Provides tools necessary for future instrument-specific monitoring/calibration scripts

Module/Tool Name	Description
utils.py	Various utility functions (e.g. filename parser, config file parser)
permissions.py	UNIX permission management for JWQL-related files
preview_image.py	Creates preview images for JWST FITS files for web application
logging_functions.py	Logs execution of scripts
monitor_mast.py	Monitor the contents of the MAST database
filesystem_monitor.py	Monitor the contents of the MAST data cache
monitor_template.py	Provides examples of how to build an instrument monitor
jwql_webapp	Django web application
pytest unit tests	Tests for individual pieces of code
Jenkins Continuous Integration	Automatically runs tests when new code is introduced to repo
sphinx API Documentation	Automatically generates code documentation

Web App



User interface
to JWQL
system

Home - JWST Quicklook <https://dijwql.stsci.edu/jwql/>

JWST QUICKLOOK HOME ABOUT DASHBOARD FGS ▾ MIRI ▾ NIRCAM ▾ NIRISS ▾ NIRSPEC ▾

WELCOME TO THE JWST QUICKLOOK PAGE.

Dashboard Query Database

The James Webb Quicklook Application (JWQL) is a database-driven web application and automation framework for use by the JWST instrument teams to monitor the health and stability of the JWST instruments. Visit our [about page](#) to learn more about our project, goals, and developers.

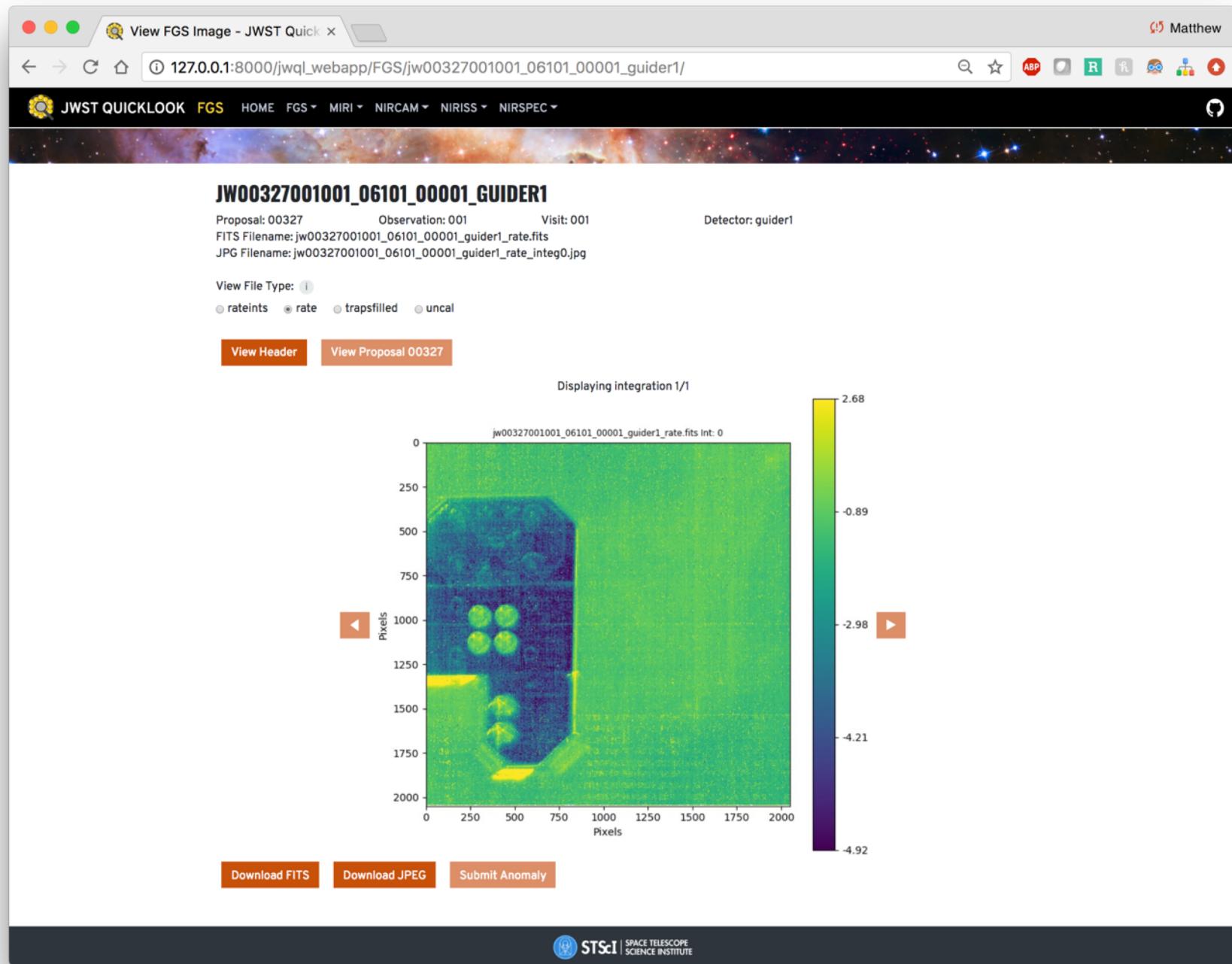
The JWQL application is currently under heavy development. The 1.0 release is expected in 2019.

STScI | SPACE TELESCOPE SCIENCE INSTITUTE

Web App



User interface
to JWQL
system



Automation



JWST
Instrument
monitoring/
calibration
software
executed daily

- Bias Monitor
- Readnoise Monitor
- Gain Level Monitor
- Dark Current Monitor
- Hot Pixel Monitor
- Photometric Stability Monitor
- Dead/Bad Pixel Population Monitor
- Flat Field Monitor
- Interpixel Capacitance/Crosstalk Monitor
- Snowball Monitor
- Wide Field Slitless Dispersion Calibration Monitor
- Pedestal Monitor
- 1/f Noise Monitor
- Average Closure Phase of AMI Calibrators Monitor
- RMS for the derivative of TSO Observation Monitor
- Engineering Database Mnemonics
- Sky/telescope Background Monitor
- PSF Characterization
- Photometric Calibration
- Internal Flats Monitor
- Target Acquisition Offsets/Failure Monitor
- Fringe Phases/Amplitudes AMI observation Monitor
- WCS Outputs Monitor
- Optical Short Monitor
- Saturation Level Monitor
- Reference Pixel Monitor
- Internal Lamp Monitor
- Instrument Model Updates
- Failed-open Shutter Monitor
- MSA Contrast
- Detector Monitoring via Fixed-slit Subarrays
- Readout Transition Monitor
- BOTS Monitoring
- Line Spread Function Monitor
- IFU Leakage Characterization
- Partial Calibration
- Cosmic Ray Monitoring
- Blind Pointing Accuracy
- Filter and Calibration Lamp Monitor
- Telescope Thermal Emission Monitoring
- MRS Straylight Monitor
- MRS Astrometric Monitor
- Geometric Distortion Monitor
- FGS-SI Alignment I Monitor
- FGS-SI Alignment II Monitor
- Focus Monitor

Automation



**JWST
Instrument
monitoring/
calibration
software
executed daily**

List of monitors can be viewed/updated at:

<https://confluence.stsci.edu/display/JWQLPROJ/Instrument+Monitor+Overview>



jwql@stsci.edu



<https://github.com/spacetelescope/jwql>



JWQLPROJ