The Hubble Space Telescope Advanced Camera for Surveys Quicklook Application

Matthew Bourque,[1] Sara Ogaz,[1] Alex Viana,[2], Meredith Durbin,[3] and Norman Grogin,[1]

[1]*Space Telescope Science Institute, Baltimore, Maryland, USA*

[2]*Terbium Labs, Baltimore, Maryland, USA*

[3]*University of Washington, Seattle, Washington, USA*

Abstract. The Hubble Space Telescope (HST) Advanced Camera for Surveys (ACS) instrument has been acquiring thousands of images each year since its installation in 2002 (via Servicing Mission 3B) and subsequent restoration in 2009 (via Servicing Mission 4). The ACS Quicklook project (`acsql`) provides a means for users to view all on-orbit ACS image and header data through a browser interface, an API to build custom datasets through relational queries, and a platform on which to build automated instrument calibration and monitoring routines. This is accomplished through the various system components: (1) a ~40 TB filesystem, which stores all publicly-available ACS data on disk, (2) a MySQL database, which stores all publicly-available FITS header data for each ACS filetype and header extension, (3) a Python/Flask-based web application, which allows users to navigate and display any public ACS archival data, and (4) a Python code library, which serves as the filesystem, database, and website API. ACS Quicklook is intended to be extended to support the forthcoming James Webb Space Telescope mission.

1. Introduction

The ACS Quicklook Application (hereinafter referred to as "`acsql`") is a Python-based application for discovering, viewing, and querying all publicly-available ACS data. It consists of: (1) A filesystem that stores ACS instrument data files and "Quicklook" JPEGs in an organized Network File System (NFS) (hereinafter referred to as the "`acsql` filesystem"), (2) A `MySQL` database that stores observational metadata of each observation (hereinafter referred to as the "`acsql` database"), (3) A Python/`Flask`-based web application for interacting with the filesystem and database (hereinafter referred to as the "`acsql` web application"), and (4) A Python code library that contains software for connecting to the database, ingesting new data, logging production code execution, and building/maintaining the web application (hereinafter referred to as the "`acsql` library" or "`acsql` package"). Below, we futher describe each of these main components.

2. Filesystem

The `acsql` filesystem is a Network File System (NFS) that stores all publically available on-orbit ACS data on disk in an organized set of directories and subdirectories hosted at STScI. Figure 1 shows an example of this directory structure. The parent directory is the first four characters of the 9-character rootname, which has a one-to-one correspondence with an individual ACS proposal Ed Smith et al. (2011). The subdirectories of the parent directories are named after the full 9-character rootname such that each parent directory contains all rootnames that were observed for the particular proposal. Furthermore, each rootname subdirectory contains every available filetype for the particular observation. Figure 1 also shows how the filesystem has evolved over the lifetime of the ACS mission based on the number of ACS observations taken from its three detectors: Wide Field Channel (WFC), High Resolution Channel (HRC) and Solar Blind Channel (SBC). Currently, the filesystem occupies ~40 TB of storage space across ~215,000 observations.

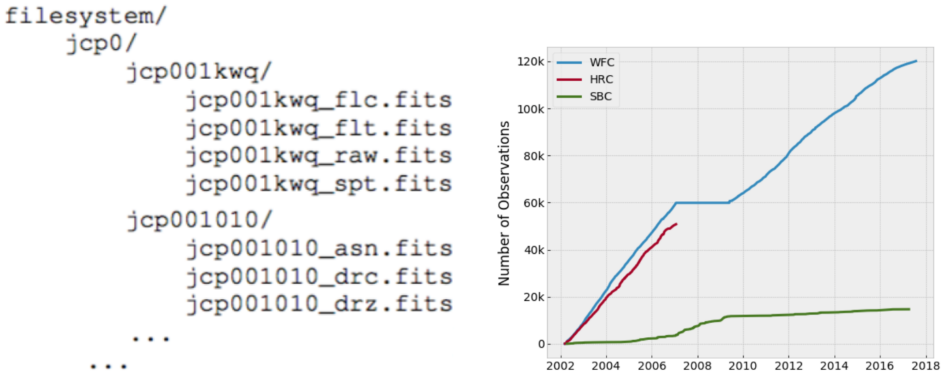

Figure 1. *Left:* A representation of the `acsql` filesystem directory structure, organized by rootname. *Right:* The number of ACS observations over time for each of the three detectors. Each observation corresponds to a single 9-character rootname.

3. Database

Another major component of the `acsql` application is a relational database that stores all FITS header key/value pairs for each ACS filetype and each FITS file extension for all on-orbit ACS observations. Such a database allows users to perform relational queries for any observational metadata based on the header keywords. To accomplish this, we implemented the relational schema shown in Figure 2. The `acsql` database contains 111 tables in total: one `master` table, which contains basic information about each rootname that is important for maintaining the database, one `datasets` table which indicate which filetypes are available for a particular rootname, and 109 'header' tables which stores the header key/value pairs, one for each `detector-filetype-extension` combination (e.g. `wfc_raw_0`).

Figure 2. *Left:* The relational schema for the `acsql` database. *Right:* An example of a query performed on the `acsql` database using the `database_interface.py` module.

4. Web Application

The front-end of the `acsql` application is the web application. The web application is built using Python and `Flask`, which is a Python based web framework (Armin Ronacher et al. (10)). Currently, the web application has two main features/modes of use: (1) viewing JPEGs and image metadata for any publicly-available ACS `raw`, `flt`, and `flc` image (when applicible), and (2) performing relational queries on the `acsql` database. To illustrate these features, we show examples of some of the different webpages that make up the web application in Figure 3 and Figure 4.

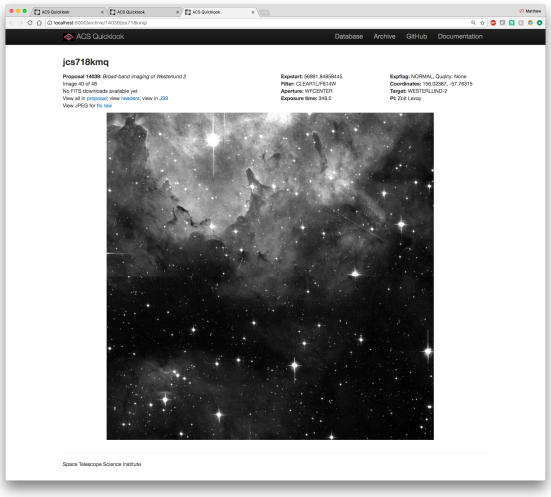

Figure 3. An example webpage for viewing dataset `jcs718kmq`.

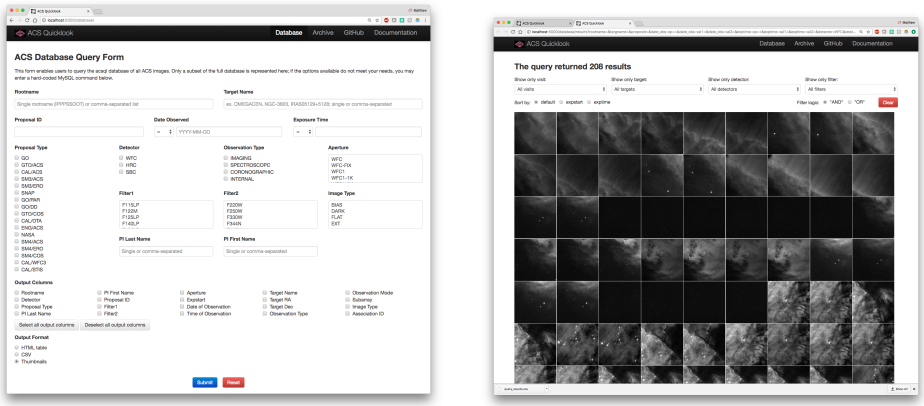

Figure 4. The Quicklook homepage (*top*), a Quicklook webpage showing results of a database query (*middle*), and a Quicklook webpage for viewing a single image (*bottom*).

Acknowledgments. We would like to thank the Advanced Camera for Surveys team at STScI for their continued support of this project. We would also like to thank the members of the Operations and Engineering Division (OED) and the Information Technology Services Division (ITSD) at STScI for their assistance on archival and computer support issues. Finally, we would like to thank the Towson University Computer Science Department for their support of this project as part of the author's Computer Science Master's Degree project.

References

Armin Ronacher, et al. 2010–, Flask: web development one drop at a time. [Online; accessed 2017-09-04], URL `http://flask.pocoo.org/`

Ed Smith, et al. 2011, Introduction to the HST Data Handbooks, 8