STScI | SPACE TELESCOPE SCIENCE INSTITUTE

JWST TECHNICAL REPORT

Title: pysiaf, a Python package for the interpretation, generation, and maintenance of space telescope Science Instrument Aperture Files (SIAF)	Doc #: JWST-STScI-007030, SM-12 Date: 8 November 2019 Rev: -
Authors: Johannes Sahlmann Phone: 410-338-2407	Release Date: 17 December 2019

1 Abstract

A space telescope Science Instrument Aperture File (SIAF) contains the detailed geometric focal plane descriptions and relationships that are necessary for science operations. For HST and JWST these files are part of the Project Reference Database (PRD). pysiaf is a public python package to access, interpret, maintain, and generate SIAF products and content (https://github.com/spacetelescope/pysiaf). This package currently supports JWST fully and HST partially, i.e. without generation capability, and it can easily be expanded to serve other missions, e.g. WFIRST, or ground-based observatories. For JWST, pysiaf replaces the previous maintenance scheme that was based on Excel spreadsheets. The package provides tools for accessing SIAF parameters, applying frame transformations, visualizing apertures, and for the comparison, verification, and validation of SIAF products. It was designed to be a robust, traceable, and user-friendly tool during pre-launch, commissioning, and operations. This document provides a high-level description of the package, its current capabilities, and the demonstrated and anticipated operational scenarios.

Table of Contents

2 Introduction

Historically, the HST SIAF has been maintained by the STScI Telescopes Branch on the basis

of Excel spreadsheets. This model was initially adopted for JWST as well and 5 Excel spreadsheets, one per JWST instrument plus the FGS, were maintained by Colin Cox and collaborators based on the inputs from the instrument and telescope teams. These Excel files were fed as input into a Java tool that produced the XML products for inclusion in the Science Operations Center (SOC) partition of the Project Reference Database (PRD), i.e. in the JWST PRDOPSSOC. Although some data manipulations had already been implemented in Python as part of the SIAF update and generation process, the procedure was labor-intensive and would have been difficult to maintain during JWST commissioning.
We therefore decided to move towards a more robust, transparent, and maintainable implementation using a public Python package hosted on GitHub and named pysiaf (https://github.com/spacetelescope/pysiaf).
The structure and scope of this package were developed in 2017—2019, where the existing Python scripts were refactored and integrated when applicable. The usability and benefits of pysiaf were demonstrated with several SIAF update deliveries to the PRD and during JWST commissioning rehearsals. This technical report describes the main characteristics and functionalities of the package. We expect that minor developments and improvements will be implemented continuously when the need arises, in addition to the regular updates that are synchronized with SOC PRD releases.

2.1 JWST instruments

JWST's payload comprises 4 science instruments (SIs), which are MIRI, NIRCam, NIRISS, and NIRSpec, plus the Fine Guidance Sensors (FGS). The main apertures are shown in Figure 1. There are thus 5 instances of JWST SIAF products which are xml files with the naming convention [SI-Name]_SIAF.xml.

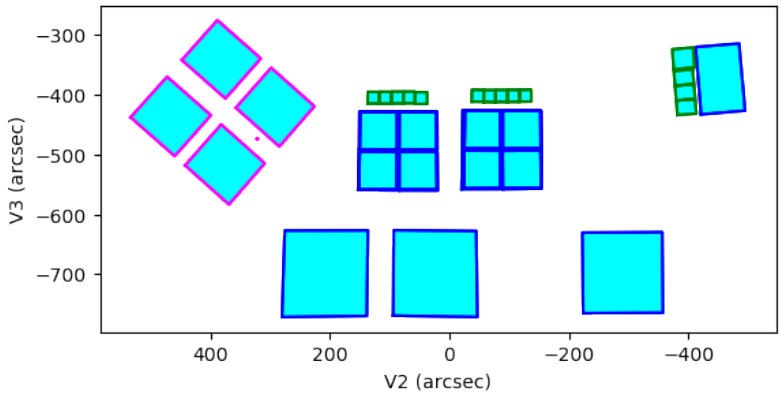


Figure 1: The main JWST apertures displayed in the telescope frame (V2, V3). This figure can be created with two lines of code:
from pysiaf.siaf import plot_main_apertures; plot_main_apertures();

2.2 JWST SIAF overview

The primary conceptual document describing the JWST SIAF is Cox & Lallo 2017 (JWST-STScI-001550). The format of the xml products is defined in the PRD Interface Requirements and Control Document (IRCD, latest version, e.g. Groebner 2017).
In broad terms, the SIAF defines the names, types and sizes of apertures for a specific instrument. It defines the locations and orientations of JWST apertures in four coordinates systems, often referred to as `frames`: Detector Frame – det, Science Frame – sci (same as

DMS coordinate system), Ideal Frame – idl (distortion-corrected system), and Telescope Frame or V(ehicle)-Frame – tel, which relates to the telescopes focal plane. The SIAF also captures the coefficients necessary for implementing transformations between these coordinate systems. In addition, subsystem-specific information is carried, e.g. the name of the aperture to be used for differential distortion correction (DDC) or the use-after date relevant for the Data Management System (DMS)[1].
The names of the SIAF fields are defined in the IRCD and they are identical for all instruments. In some cases, fields were overloaded to allow for instrument-specific usages, e.g. for some NIRSpec transformations (Proffitt et al. 2017). This means that although the xml tag may be the same for two instruments, the value of that field may not be used in the same way.
Figure 2 shows an attempt to capture the flow of SIAF content through the JWST Science and Operations Center (S&OC). The SIAF has many customers within the S&OC and outside, since the public pysiaf package is included as dependency for independent software packages like WebbPSF and mirage.

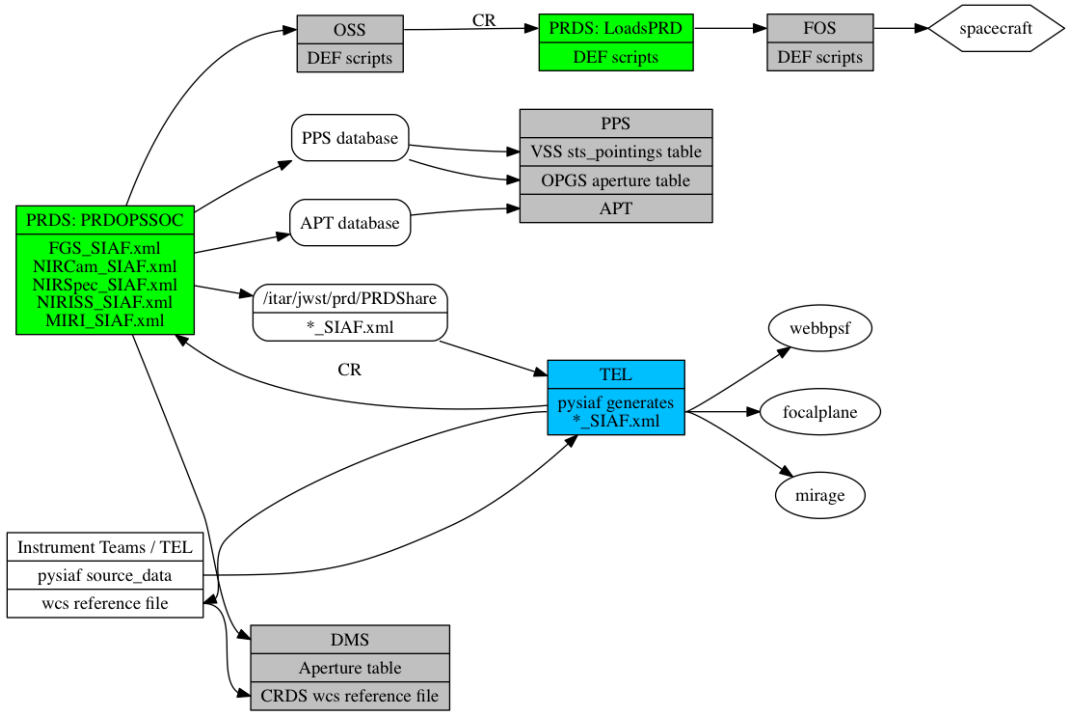


Figure 2: The flow of SIAF information in the JWST S&OC. The yellow boxes refer to PRD partitions. The Python code to produce this figure can be found at https://github.com/spacetelescope/astrometry-scripts/blob/master/siaf_related/jwst_siaf_flow_graph.py.

3 Design and structure of pysiaf

In this section we discuss the principles and structure that have guided us through the implementation of pysiaf.

[1] A UseAfterDate of YYYY-MM-DD is interpreted by DMS as YYYY-MM-DDT00:00:00 when it performs time-dependent queries (M. Swam, 2018, private communication).

3.1 Open source Python package

We chose the Python programming language because it is the dominant programming language for the JWST SOC and it is widely used in astronomy and astrophysics. We host pysiaf on GitHub's spacetelescope organization page and made it public, because the SIAF is not a restricted-access product and because publicly-available software delivered as installable package can easily be added as dependency in other software projects. In the following we list noteworthy features of pysiaf:

- pysiaf includes an up-to-date copy of the SIAF xml files as present in the SOC PRD. This, in principle, enables a single public access point to SIAF information which is controlled by STScI.
- We develop pysiaf using STScI's best practices as much as possible, e.g. in terms of code style and documentation, using pull-request reviews by peers, implementation of test suites and continuous integration and delivery via PyPI and astroconda.
- Bug reports and feature requests can be made via github tickets, and we integrated pysiaf with the internally-used JWSTSIAF JIRA project.
- The code documentation is available at https://pysiaf.readthedocs.io/en/latest/
- pysiaf has a digital object identifier (DOI) and can be cited as Sahlmann et al. (2019b), see https://doi.org/10.5281/zenodo.3517245.

3.2 Release and update git workflow

Hosting pysiaf on GitHub comes with the in-built features of version control and software releases. We eventually adopted a forking workflow, where the feature and release branches are on the forks whereas the master and develop branches are on the upstream repository.

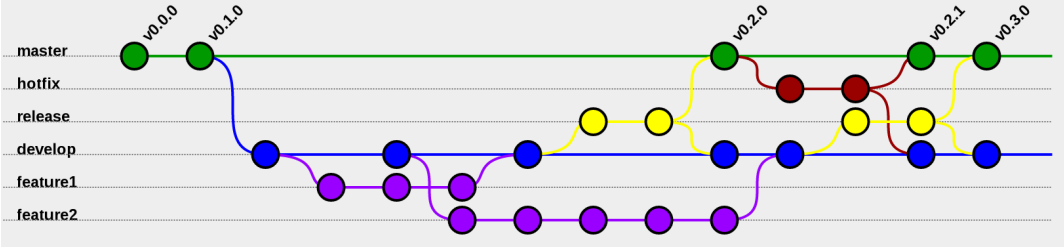


Figure 3: Visualization of the pysiaf git workflow, which follows the model of https://github.com/spacetelescope/jwql

Following the workflow shown in Figure 3 code changes are introduced in feature branches and accumulate in the develop branch until a new release is drafted. To accomplish a release two pull requests against the master and the develop branch are made such that these branches are even at the time of the release. Releases are published as needed through GitHub and are delivered automatically to PyPI via the Travis build. Updating the pysiaf version available through Astroconda necessitates human intervention and
more detailed documentation is available under the *Procedures* tab at
https://innerspace.stsci.edu/display/INSTEL/JWST+SIAF.

3.3 Generation scripts

There are 5 top-level scripts that are located in the /generate folder of the repository, one per SI plus FGS. These files are named generate_[SI-NAME].py and generate the SIAF xml product as well as diagnostic products (difference files, figures, and tables) when executed. Their only

inputs are the respective source data.
When generating the SIAF content, these scripts also perform various automated checks for consistency and formatting of the aperture parameters.
By design principle, the generation scripts on the pysiaf master branch reproduce the latest SOCPRD release of the SIAF files exactly. This may not necessarily be the case on the develop branch which is intended to be used for implementing SIAF updates.

3.4 The Source Data

We refer to the source data as the minimal set of input files necessary to generate the SIAF xml product. Typically, these data are delivered by the instrument team to the SIAF maintainers. The interface is defined in terms of the structure and naming of products in pysiaf's /source_data directory (see e.g. https://jira.stsci.edu/browse/JWSTSIAF-128 for NIRSpec or https://jira.stsci.edu/browse/JWSTSIAF-136 for MIRI), such that updates to the source data can be delivered through a pull request in the GitHub repository.
In this way, the generation script remains unchanged and any change due to an update is limited to the source data which can easily be verified and tracked through the git version control.
For some instruments, testing datasets are included in the source data and in these cases the pysiaf generation scripts implement automated regression tests against the instrument team's independent results. This improves reliability and helps to catch potential issues early.

The following sections describe some of the standardized source data components in more detail.

3.4.1 Detector layout (siaf_detector_layout.txt)

This file (Figure 4) compiles information for all SI sensor chip assemblies (SCAs), i.e. the instrument name, main aperture name, orientation and parity between Detector and Science frame, and the parity between Ideal and Telescope Frames.

```
# SIAF detector layout definition file.
#
# These apertures act as parent apertures of all other SI apertures and their parameters are thus inherited.
#
# Generated 2018-04-19T16:44:22.903 utc
# jsahlmann
#
InstrName ,   AperName , DetSciYAngle , DetSciParity , VIdlParity
   NIRCAM , NRCA1_FULL ,            0 ,           -1 ,         -1
   NIRCAM , NRCA3_FULL ,            0 ,           -1 ,         -1
   NIRCAM , NRCA5_FULL ,            0 ,           -1 ,         -1
   NIRCAM , NRCB2_FULL ,            0 ,           -1 ,         -1
   NIRCAM , NRCB4_FULL ,            0 ,           -1 ,         -1
   NIRCAM , NRCA2_FULL ,          180 ,           -1 ,         -1
   NIRCAM , NRCA4_FULL ,          180 ,           -1 ,         -1
   NIRCAM , NRCB1_FULL ,          180 ,           -1 ,         -1
   NIRCAM , NRCB3_FULL ,          180 ,           -1 ,         -1
   NIRCAM , NRCB5_FULL ,          180 ,           -1 ,         -1
      FGS ,  FGS1_FULL ,          180 ,            1 ,         -1
      FGS ,  FGS2_FULL ,            0 ,           -1 ,         -1
   NIRISS ,    NIS_CEN ,          180 ,            1 ,         -1
  NIRSPEC ,  NRS1_FULL ,            0 ,            1 ,         -1
  NIRSPEC ,  NRS2_FULL ,          180 ,            1 ,         -1
     MIRI , MIRIM_FULL ,            0 ,            1 ,         -1
```

Figure 4: Content of the detector layout source data file (siaf_detector_layout.txt).

3.4.2 XML field format (siaf_xml_field_format.txt)

This file (Figure 5) specifies the format of each SIAF XML tag in compliance with the IRCD. The numerical precision of each field is also specified although that is not part of the IRCD and no explicit requirements exist.

Until PRDOPSSOC-M-025 the numerical precisions were heterogeneous across instruments and one field (DetSciYAngle) was not IRCD compliant (see https://jira.stsci.edu/browse/JWSTSIAF-120). This was resolved and precisions are homogenous starting with PRDOPSSOC-M-026. At the same time, the numerical resolution of angles given in degrees was increased to match the expected precision of calibrations that utilize the high-accuracy Gaia reference frame.

```
# xml field format reference file for SIAF
#
# This file describes the format and order of the XML fields present in the SIAF.
# This is compliant with the IRCD (JWST-STScI-000949) and complementary because the decimal
# precision of float fields is specified.
#
# Generated 2018-02-20T02:00:15.117 utc by Johannes Sahlmann
#
# Updated 2018-07-30 by Johannes Sahlmann:
# Adapted from NIRSpec for uniform formatting across instruments
#
field_nr ,           field_name ,     format ,   pyformat ,                        example
     1.1 ,            InstrName ,     string ,            ,                        NIRSPEC
     1.2 ,             AperName ,     string ,            ,                      NRS1_FULL
     1.3 ,              DDCName ,     string ,            ,                      NRS1_CNTR
     1.4 ,             AperType ,     string ,            ,                        FULLSCA
     1.5 ,            AperShape ,     string ,            ,                           QUAD
     1.6 ,             XDetSize ,    integer ,          d ,                           2048
     1.7 ,             YDetSize ,    integer ,          d ,                           2048
     1.8 ,              XDetRef ,      float ,        .2f ,                        1024.50
     1.9 ,              YDetRef ,      float ,        .2f ,                        1024.50
    1.10 ,             XSciSize ,    integer ,          d ,                           2048
    1.11 ,             YSciSize ,    integer ,          d ,                           2048
    1.12 ,              XSciRef ,      float ,        .2f ,                        1024.50
    1.13 ,              YSciRef ,      float ,        .2f ,                        1024.50
    1.14 ,            XSciScale ,      float ,        .8f ,                     0.10326748
    1.15 ,            YSciScale ,      float ,        .8f ,                     0.10532422
    1.16 ,                 V2Ref ,      float ,        .6f ,                     294.054813
    1.17 ,                 V3Ref ,      float ,        .6f ,                    -502.901911
    1.18 ,          V3IdlYAngle ,      float ,        .8f ,                     139.155450
    1.19 ,           VIdlParity ,    integer ,          d ,                             -1
    1.20 ,         DetSciYAngle ,    integer ,          d ,                              0
    1.21 ,         DetSciParity ,    integer ,          d ,                              1
    1.22 ,          V3SciXAngle ,      float ,        .8f ,                      48.133502
    1.23 ,          V3SciYAngle ,      float ,        .8f ,                     139.155450
    1.24 ,            XIdlVert1 ,      float ,        .4f ,                       -105.5208
    1.25 ,            XIdlVert2 ,      float ,        .4f ,                        104.1228
    1.26 ,            XIdlVert3 ,      float ,        .4f ,                        106.6586
    1.27 ,            XIdlVert4 ,      float ,        .4f ,                       -107.4969
    1.28 ,            YIdlVert1 ,      float ,        .4f ,                       -104.9906
    1.29 ,            YIdlVert2 ,      float ,        .4f ,                       -108.0142
    1.30 ,            YIdlVert3 ,      float ,        .4f ,                        107.6142
    1.31 ,            YIdlVert4 ,      float ,        .4f ,                        111.4559
    1.32 ,         UseAfterDate ,     string ,            ,                      2014-01-01
    1.33 ,              Comment ,     string ,            ,
    1.34 ,            Sci2IdlDeg ,    integer ,          d ,                              5
    1.35 ,            Sci2IdlX00 ,      float ,       .15e ,          -6.766024243397535E-02
    1.36 ,            Sci2IdlX10 ,      float ,       .15e ,           6.371061788272568E-05
    1.37 ,            Sci2IdlX11 ,      float ,       .15e ,          -4.967884326775753E-07
```

Figure 5: Content of the xml format source data file (siaf_xml_field_format.txt).

3.4.3 Aperture definition (nircam_siaf_aperture_definition.txt)

This fundamental file contains a table that defines the apertures to be included for the respective instrument. The order of listed apertures is preserved and for any aperture this file

defines the aperture name and type, the X,Y pixel coordinates of the reference position in Detector and Science Frames, the aperture size in pixels in the Science Frame, parent aperture(s), and the dependency type (see Section 3.5). This file also has a header for auxiliary information, in particular explicit modification notes and dates. An example is shown in Figure 6.

```
# NIRCam aperture definition file for SIAF
#
# This file contains all the necessary aperture information to generate the full SIAF given the necessary reference files (focal plane alignment, distortion) and
  auxiliary information (DDC mapping, wedge offsets, ...)
# This file also defines the order in which the apertures are presented.
#
# Originally based on PRDOPSSOC-H-015.
#
# Generated 2018-06-12T13:57:29.020 utc
# jsahlmann
#
# jsahlmann 2018-06-12:
# Modified XDetRef,YDetRef of COMPOUND apertures NRCAS_FULL, NRCBS_FULL, NRCALL_FULL to be able
# to derive the V2,V3 reference point from a pixel coordinate on a detector.
#
# jsahlmann 2018-07-26:
# Modified XSciRef,YSciRef of 6 apertures to match half-integer values of DetRef
#
# jstans, Gennaro 2019-10-07
# Added NRCA5_TAGRISMTS32_F405N to support TA in the F405N filter as well as F335M. This is for future use, not currently supported.
#   Note that the XDetRef and YDetRef value has NOT been corrected for filter-dependent wander. See JWSTSIAF-159.
# Added 12 FULL apertures corresponding to the Coronagraphic TA apertures. These support calibration of the astrometric confirmation
#   images that PRECEDE the SAM from the TA aperture to the coronagraphic mask. See JWSTSIAF-61.
#
                AperName , AperType , XDetRef , YDetRef , XSciSize , YSciSize , XSciRef , YSciRef ,                                parent_apertures , dependency_type
          NRCA1_FULL_OSS ,      OSS ,  1024.5 ,  1024.5 ,     2048 ,     2048 ,  1024.5 ,  1024.5 ,                                      NRCA1_FULL ,          default
          NRCA2_FULL_OSS ,      OSS ,  1024.5 ,  1024.5 ,     2048 ,     2048 ,  1024.5 ,  1024.5 ,                                      NRCA2_FULL ,          default
          NRCA3_FULL_OSS ,      OSS ,  1024.5 ,  1024.5 ,     2048 ,     2048 ,  1024.5 ,  1024.5 ,                                      NRCA3_FULL ,          default
          NRCA4_FULL_OSS ,      OSS ,  1024.5 ,  1024.5 ,     2048 ,     2048 ,  1024.5 ,  1024.5 ,                                      NRCA4_FULL ,          default
          NRCA5_FULL_OSS ,      OSS ,  1024.5 ,  1024.5 ,     2048 ,     2048 ,  1024.5 ,  1024.5 ,                                      NRCA5_FULL ,          default
          NRCB1_FULL_OSS ,      OSS ,  1024.5 ,  1024.5 ,     2048 ,     2048 ,  1024.5 ,  1024.5 ,                                      NRCB1_FULL ,          default
          NRCB2_FULL_OSS ,      OSS ,  1024.5 ,  1024.5 ,     2048 ,     2048 ,  1024.5 ,  1024.5 ,                                      NRCB2_FULL ,          default
          NRCB3_FULL_OSS ,      OSS ,  1024.5 ,  1024.5 ,     2048 ,     2048 ,  1024.5 ,  1024.5 ,                                      NRCB3_FULL ,          default
          NRCB4_FULL_OSS ,      OSS ,  1024.5 ,  1024.5 ,     2048 ,     2048 ,  1024.5 ,  1024.5 ,                                      NRCB4_FULL ,          default
          NRCB5_FULL_OSS ,      OSS ,  1024.5 ,  1024.5 ,     2048 ,     2048 ,  1024.5 ,  1024.5 ,                                      NRCB5_FULL ,          default
             NRCALL_FULL , COMPOUND ,  2713.5 ,  2057.5 ,     None ,     None ,    None ,    None , NRCA1_FULL; NRCB2_FULL; NRCB1_FULL; NRCA2_FULL , nircam_compound
              NRCAS_FULL , COMPOUND ,  1964.0 ,  1978.5 ,     None ,     None ,    None ,    None , NRCA1_FULL; NRCA3_FULL; NRCA4_FULL; NRCA2_FULL , nircam_compound
              NRCA1_FULL ,  FULLSCA ,  1024.5 ,  1024.5 ,     2048 ,     2048 ,  1024.5 ,  1024.5 ,                                            None ,          master
              NRCA2_FULL ,  FULLSCA ,  1024.5 ,  1024.5 ,     2048 ,     2048 ,  1024.5 ,  1024.5 ,                                            None ,          master
              NRCA3_FULL ,  FULLSCA ,  1024.5 ,  1024.5 ,     2048 ,     2048 ,  1024.5 ,  1024.5 ,                                            None ,          master
              NRCA4_FULL ,  FULLSCA ,  1024.5 ,  1024.5 ,     2048 ,     2048 ,  1024.5 ,  1024.5 ,                                            None ,          master
              NRCA5_FULL ,  FULLSCA ,  1024.5 ,  1024.5 ,     2048 ,     2048 ,  1024.5 ,  1024.5 ,                                            None ,          master
              NRCBS_FULL , COMPOUND ,    87.0 ,  1974.0 ,     None ,     None ,    None ,    None , NRCB4_FULL; NRCB2_FULL; NRCB1_FULL; NRCB3_FULL , nircam_compound
              NRCB1_FULL ,  FULLSCA ,  1024.5 ,  1024.5 ,     2048 ,     2048 ,  1024.5 ,  1024.5 ,                                            None ,          master
              NRCB2_FULL ,  FULLSCA ,  1024.5 ,  1024.5 ,     2048 ,     2048 ,  1024.5 ,  1024.5 ,                                            None ,          master
              NRCB3_FULL ,  FULLSCA ,  1024.5 ,  1024.5 ,     2048 ,     2048 ,  1024.5 ,  1024.5 ,                                            None ,          master
              NRCB4_FULL ,  FULLSCA ,  1024.5 ,  1024.5 ,     2048 ,     2048 ,  1024.5 ,  1024.5 ,                                            None ,          master
              NRCB5_FULL ,  FULLSCA ,  1024.5 ,  1024.5 ,     2048 ,     2048 ,  1024.5 ,  1024.5 ,                                            None ,          master
            NRCB1_FULLP ,  FULLSCA , 1430.94 ,  675.99 ,     2048 ,     2048 , 1430.94 , 1373.01 ,                                      NRCB1_FULL ,          default
```

Figure 6: Content (truncated) of the NIRCam aperture definition source data file (nircam_siaf_aperture_definition.txt)

3.4.4 Detector parameters ([SI-name]_siaf_detector_parameters.txt)

This file specifies the SI-specific detector/SCA pixel dimensions, see Figure 7.

```
# MIRI detector parameter definition file for SIAF
#
# This file contains the basic detector characteristics.
#
# Generated 2018-04-11T00:55:34.126 utc
# jsahlmann@nereus.local.stsci.edu
#
InstrName , AperShape , XDetSize , YDetSize
     MIRI ,      QUAD ,     1032 ,     1024
```

Figure 7: Content of the MIRI detector parameter source data file (miri_siaf_detector_parameters.txt)

3.4.5 Alignment parameters ([SI-name]_siaf_alignment.txt)

These reference files capture the calibration parameters that are determined from the FGS-to-SI alignment observation programs, i.e. the locations and orientations of the main apertures. These fundamental parameters are V2Ref, V3Ref, and V3IdlYAngle (see Cox & Lallo 2017). Since these parameters are determined for every detector individually, there can be multiple

alignment parameter files for one instrument. An example is shown in Figure 8.

```
# NIRCam alignment parameter reference file for SIAF
#
# This file contains the focal plane alignment parameters of master apertures calibrated
# during FGS-SI alignment.
#
# Generated 2018-07-26T13:35:21.437 utc
# by jsahlmann
#
                    AperName ,          V3IdlYAngle ,         V3SciXAngle ,          V3SciYAngle ,              V2Ref ,               V3Ref
                  NRCA1_FULL ,  -0.5627897984331705 , -90.51540196741483 ,  -0.5627897984331705 ,  120.52546407006393 ,  -527.5431321986374
                  NRCA2_FULL , -0.21969951947477284 ,  -90.2774540548771 , -0.21969951947477284 ,   119.5890989791352 , -459.86330405774663
                  NRCA3_FULL ,  0.27380526727242016 , -89.96823541000089 ,  0.27380526727242016 ,   51.88558248294482 ,  -527.9527524294305
                  NRCA4_FULL ,  0.08871159323578581 , -89.95986089758783 ,  0.08871159323578581 ,   52.19455017805899 , -459.97249016971165
                  NRCA5_FULL , -0.07385126575571908 ,  -90.1569305723844 , -0.07385126575571908 ,   86.03901141872866 ,  -493.3857038417111
                  NRCB1_FULL ,   0.3665986996931565 , -89.57306239178865 ,   0.3665986996931565 , -120.96951292849471 ,  -457.7887772274396
                  NRCB2_FULL ,    0.85955164889866 , -89.17522936742581 ,    0.85955164889866 , -121.15468437050728 ,  -525.4781411837679
                  NRCB3_FULL , -0.43660166890663277 , -90.39856852606466 , -0.43660166890663277 ,  -53.13993446339103 , -457.82173997940043
                  NRCB4_FULL ,  -0.3168049798862335 , -90.07768790555797 ,  -0.3168049798862335 ,  -52.84570250197705 ,  -525.7293209330118
                  NRCB5_FULL , 0.008057715335468174 , -89.91498299353484 , 0.008057715335468174 ,  -89.59651439128646 ,   -491.474153244519
```

Figure 8: Content of the NIRCam alignment parameter source data file (nircam_siaf_alignment.txt)

3.4.6 Geometric distortion coefficients ([SI-name]_ siaf_distortion_[aperture-name].txt)

These files contain the bivariate polynomial coefficients that map the geometric distortion of a main aperture in both directions. The exponents are specified and since the polynomial degree differs between instruments, the number of rows can vary. There is one such file per detector. An example is shown in Figure 9.

The content of these files will be updated based on in-flight distortion calibration observations.

```
# NIRISS distortion reference file for SIAF

# Aperture: NIS_CEN
# Based on coefficients from Martel & Fullerton "The Geometric Distortion of NIRISS" (JWST-STScI-003524 Rev A),
# that were rescaled, shifted for a different reference pixel location, and rotated:
# Rotation of 0.570 deg was removed and is carried separately in V3IdlYangle.
# These parameters are stored in PRDOPSSOC-H-014.
#
# Generated 2018-02-21T01:33:49.873 utc
# jsahlmann
#
AperName , siaf_index , exponent_x , exponent_y ,               Sci2IdlX ,               Sci2IdlY ,               Idl2SciX ,               Idl2SciY
 NIS_CEN ,         00 ,          0 ,          0 ,                   -0.0 ,                    0.0 ,                    0.0 ,                    0.0
 NIS_CEN ,         10 ,          1 ,          0 ,     0.06539229315982802 ,  -0.0002985551192185085 ,      15.29222837186319 ,     0.06939322901426978
 NIS_CEN ,         11 ,          0 ,          1 , -1.0842021724855044e-19 ,     0.06588855460670805 ,  -7.564810278568035e-05 ,      15.177308244392062
 NIS_CEN ,         20 ,          2 ,          0 ,  2.3048392287757872e-08 ,  -4.953440669097896e-08 ,  -8.159831790274738e-05 ,  0.0001751031215762744 6
 NIS_CEN ,         21 ,          1 ,          1 ,  -5.376986687030241e-09 ,  2.7241638468616585e-08 ,  1.8740237434550763e-05 ,  -9.642680318361025e-05
 NIS_CEN ,         22 ,          0 ,          2 ,  2.5129657978351204e-09 ,  -4.193126556437237e-09 ,  -9.312167675948276e-06 ,  1.5191016636982562e-05
 NIS_CEN ,         30 ,          3 ,          0 , -2.3007847857064428e-12 , -1.3758086980395105e-12 ,  1.2927461553977667e-07 ,   6.759263333266902e-08
 NIS_CEN ,         31 ,          2 ,          1 ,  -6.313691241818919e-12 ,  -1.399882791488933e-11 ,   3.499532956150016e-07 ,   7.514091620105573e-07
 NIS_CEN ,         32 ,          1 ,          2 ,  -8.085421712640355e-12 , -1.8087349819633073e-12 ,  4.3899778410632647e-07 ,  1.2224049721822834e-07
 NIS_CEN ,         33 ,          0 ,          3 ,  -8.482262568737343e-13 ,  -2.804139706389061e-11 ,   3.769921582434318e-08 ,   1.457108515976762e-06
 NIS_CEN ,         40 ,          4 ,          0 , -1.6663765152753104e-15 ,  1.5840814839995905e-15 ,  1.3486040486136644e-09 , -1.2527827563355134e-09
 NIS_CEN ,         41 ,          3 ,          1 ,   8.187778580319582e-15 , -1.0888526590744654e-15 , -6.7088020659272854e-09 ,   7.668320387371265e-10
 NIS_CEN ,         42 ,          2 ,          2 , -1.3699613151535303e-15 ,   8.980484312743932e-15 ,  1.0598566110621749e-09 ,  -7.189867542549804e-09
 NIS_CEN ,         43 ,          1 ,          3 ,   7.305634702027793e-15 ,   2.000106773098919e-16 ,  -5.801980933129771e-09 , -1.0797767002010823e-10
 NIS_CEN ,         44 ,          0 ,          4 ,   2.573865493162265e-16 ,   1.648174517164098e-14 ,  -6.244890475361233e-11 , -1.3448939379046278e-08
```

Figure 9: Content of the NIRISS distortion source data file (niriss_siaf_distortion_nis_cen.txt)

3.4.7 DDC aperture name mapping ([SI-name]_siaf_ddc_apername_mapping.txt)

A differential distortion correction is implemented as part of the Operation Plan Generation Subsystem (OPGS). This file allows to map the SIAF aperture name to the OPGS aperture name and the DDC aperture name is assigned during the SIAF generation process based on geometric proximity (see Cox & Lallo 2017). An example is shown in Figure 10.

```
# NIRCam DDC mapping definition file for SIAF
#
# This file contains the DDC aperture mapping.
#
# Generated 2018-06-12T13:57:29.020 utc
# by jsahlmann
#
     SIAF_NAME ,     DDC_NAME
    NRCBS_FULL ,    NRCB_CNTR
   NRCALL_FULL ,  NRCALL_CNTR
NRCA5_MASK430R , NRC_MASK430R
NRCA5_MASK335R , NRC_MASK335R
 NRCA5_MASKLWB ,  NRC_MASKLWB
 NRCA4_MASKSWB ,  NRC_MASKSWB
    NRCAS_FULL ,    NRCA_CNTR
NRCA2_MASK210R , NRC_MASK210R
```

Figure 10: Content of the NIRCam DDC source data file (nircam_siaf_ddc_apername_mapping.txt)

3.4.8 Instrument-specific information

Finally, pysiaf allows for the definition of instrument-specific files such as the NIRCam grism parameters or wedge offsets, see Figure 11.

```
# NIRCam detector parameter definition file for SIAF
#
# This file contains the wedge offsets.
#
# Generated 2018-07-26T13:35:21.437 utc
# by jsahlmann
#
         name ,  v2_offset ,  v3_offset
        NRCA5 , -2.0946264 , 47.0885551
NRCA5335R430R ,  -2.121191 , 47.7105149
        NRCA4 , -1.3878951 ,  40.352564
        NRCA2 , -0.9842734 , 40.6223753
        NRCB5 ,  2.2208058 , 47.5127381
        NRCB3 ,   1.585703 , 40.1257254
        NRCB1 ,   1.765293 , 40.1731467
```

Figure 11: Coronagraphic wedge parameter of NIRCam (nircam_siaf_wedge_offsets.txt).

3.5 Handling of aperture dependencies

In many cases, the parameters of an aperture depend on the parameters of a superordinate aperture. We defined a concept of main/master and parent apertures that typically correspond to the detector/SCA and dependent apertures whose parameters depend on their parent aperture(s). A simple case is a subarray aperture which inherits its parameters from the parent main aperture.

The relationships between apertures are defined in the aperture definition file (Section 3.4.3). In addition to parent aperture information, that file specifies a *dependency_type* which allows the customize the operations implemented in the generation script.

3.6 Code documentation

The automatically generated pysiaf code documentation which describes all available classes, methods, and functions in detail can be obtained in PDF format from ReadTheDocs at https://readthedocs.org/projects/pysiaf/downloads/pdf/latest/ .

4 The package structure of pysiaf

The structure of the pysiaf package follows mostly the standard format of python packages. The top-level directories relevant for users are /generate (holds the SIAF generation scripts plus several scripts for the generation of pre-flight or general source data files), /examples (contains example scripts and python notebooks), and /pysiaf which holds the actual package content. This section presents a broad overview of the available modules and directories. Detailed code documentation is available at https://pysiaf.readthedocs.io/en/latest/.

4.1 Package directories under /pysiaf

4.1.1 /iando: Input and output

Functions to read files, e.g. JWST SIAF and HST SIAF or any of the general source data files, and to write files, in particular the JWST SIAF xml products (Excel, csv, and other astropy.table formats are also supported for the output).

4.1.2 /prd_data: copies of JWST and HST SIAF files released in the PRD

The /JWST directory contains copies of the SIAF as included in recent SOC PRD releases in XML and Excel format.
The /HST directory contains copies of recent HST SIAF files (siaf.dat) and HST FGS-specific files (amu.rep), which contain the 3D rotation matrix elements that transform between the HST FGS ideal frame and the V frame, often named `TVS matrix` (e.g. Sahlmann et al. 2019a).

4.1.3 /pre_delivery_data: used for SIAF update process

This directory contains instrument-specific sub-directories which are used to store data that is necessary for the preparation of SIAF updates and eventually PRD change requests, e.g. pre-delivery SIAF files and diagnostic tables, difference files, or figures.
This directory should only be used on the development branch (named `siaf-updates`) and it should be empty on the master branch.

4.1.4 /source_data: data for SIAF generation

This directory holds all data necessary to generate the 5 SIAF files using the generation scripts as described in Section 3.4.

4.1.5 /temporary_data

This directory should generally be empty and only be used under particular circumstances.

4.1.6 /tests: unit and system tests

A number of unit and system tests for various modules, methods, and functions are collected in this directory. They can be run locally, e.g. using pytest, and they are run by the automated Continuous Integration service, currently implemented using Travis, for every pull request, merge commit, and package release.

4.1.7 /utils: the utils module

Here we collected functions for various purposes.

- Compare.py
  Contains functions to support the comparison of SIAF products, e.g. an updated SIAF

against the PRD version of the same file, and to produce related diagnostics, e.g. comparison figures and the comparison of roundtrip transformation errors.

- Polynomial.py
  Functions to deal with bivariate polynomials and their coefficients. The original version of the module was written by Colin Cox.
- Projection.py
  Functions to support tangent-plane projection and de-projection.
- Rotations.py
  Functions to support dealing with 3D matrix rotations, e.g. the constructions of the attitude matrix and its use for transforming back and forth between equatorial coordinates (RA, Dec) and the telescope coordinate system (V2, V3). The original version of the module was written by Colin Cox.
- Tools.py
  Necessary and useful functions that did not fit into any of the other modules.

4.2 Class definitions

Pysiaf uses two important classes, the Aperture class and the ApertureCollection class.

4.2.1 The Aperture class

The pysiaf Aperture class is defined in aperture.py and holds the SIAF XML fields in its attributes. A large number of methods has been implemented to support generation, validation, and plotting and in particular to support transformations between coordinate systems.
The base class was extended through class inheritance to implement JwstAperture, NIRSpecAperture, and HstAperture, which maintains the common functionality of all these classes but allows us to define additional methods or specific method behavior as needed. For example, an JWST FGS aperture will use the 3 alignment parameters (V2Ref, V3Ref, V3IdlYAngle) to transform between the Ideal and the Telescope Frame, whereas an HST FGS aperture will use the 9 coefficients of the TVS matrix for the same transformation.

As described in Sahlmann et al. (2019a), particular care was taken to implement both the exact 3D rotation and the approximate planar/2D approximation of all relevant transformations between coordinate systems.

4.2.2 The ApertureCollection and Siaf classes

As the name suggests, the ApertureCollection class defined in siaf.py defines a container for a collection of Aperture objects and provides some useful functionality. The Siaf subclass inherits the ApertureCollection superclass.

5 Functionalities of pysiaf

5.1 Accessing SIAF parameters

The most basic use of pysiaf is to programmatically access SIAF parameters. By default, the content of the latest SOC PRD release is loaded.

```
In [15]: from pysiaf.siaf import Siaf

In [16]: s = Siaf('FGS')

In [17]: s.apernames
Out[17]: odict_keys(['FGS1_FULL_OSS', 'FGS1_FULL', 'FGS2_FULL_OSS', 'FGS2_FULL', 'FGS1_SUB128LL', 'FGS1_SUB128DIAG', 'FGS1_SU
FGS1_SUB8DIAG', 'FGS1_SUB8CNTR', 'FGS2_SUB128LL', 'FGS2_SUB128DIAG', 'FGS2_SUB128CNTR', 'FGS2_SUB32LL', 'FGS2_SUB32DIAG', 'FG
 'FGS1_FP2MIMF', 'FGS1_FP3MIMF', 'FGS1_FP4MIMF', 'FGS1_FP5MIMF', 'FGS2_FP1MIMF', 'FGS2_FP2MIMF', 'FGS2_FP3MIMF', 'FGS2_FP4MIM

In [18]: s['FGS1_FULL'].AperType
Out[18]: 'FULLSCA'

In [19]: s['FGS1_FULL'].V2Ref
Out[19]: 207.19
```

5.2 Transformations

5.2.1 Between SIAF frames

Frame transformations are available as Aperture class methods that take two coordinates as inputs. All combinations of transformations between Detector, Science, Ideal, and Telescope frames are available with a syntax similar to below:

```
In [32]: import numpy as np

In [33]: aperture = Siaf('niriss')['NIS_CEN']

In [34]: x_sci = np.linspace(1, 10, 3)

In [35]: y_sci = np.linspace(1, 10, 3)

In [36]: aperture.sci_to_tel(x_sci, y_sci)
Out[36]:
(array([-223.89539988, -224.18604762, -224.47670145]),
 array([-765.24373488, -764.94656923, -764.6493924 ]))
```

In addition, transformations to and from the `Raw` coordinate system are supported, which is the native instrument coordinate system. As a reminder, the pixel-indexing scheme of the JWST SIAF is 1-based, as opposed to the JWST pipeline's 0-based pixel indexing scheme (Cox & Lallo 2017, Law 2017)

5.2.2 Transforming to/from equatorial sky coordinates using attitude matrix

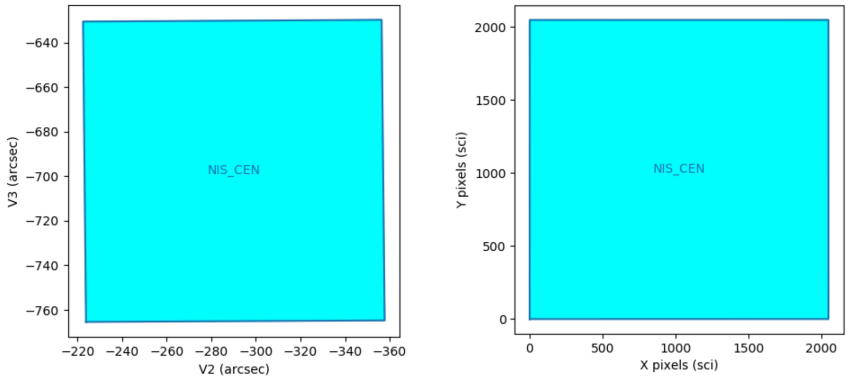
Pointing transformations between RA, Dec and V2, V3 coordinates do not depend on the aperture information and require the attitude matrix as input. Therefore these functions are available from the rotations.py module.
In the example below, we defined the attitude by a set of equatorial coordinates and position angle that map to the reference position of the aperture. We then determine the attitude matrix, apply the transformation to V2, V3, and verify that the reference position is recovered as expected.

```
In [57]: from pysiaf.utils import rotations

In [58]: import astropy.units as u

In [59]: aperture.V2Ref
Out[59]: -290.1

In [60]: aperture.V3Ref
Out[60]: -697.5

In [61]: ra_deg = 10.; dec_deg = -45.; pa_deg = 13.;

In [62]: attitude = rotations.attitude_matrix(aperture.V2Ref, aperture.V3Ref, ra_deg, dec_deg, pa_deg)

In [63]: v2, v3 = rotations.sky_to_tel(attitude, ra_deg, dec_deg)

In [64]: v2.to(u.arcsecond)
Out[64]: <Quantity -290.1 arcsec>

In [65]: v3.to(u.arcsecond)
Out[65]: <Quantity -697.5 arcsec>
```

5.3 Plotting

Both the Aperture and Siaf class have plotting methods and several functions for displaying ensembles of apertures are available as well. As an example we show the display of an aperture in the default Telescope frame and then, using one additional argument, in the Science frame. All SIAF coordinate systems are supported for plotting.

```
In [75]: aperture.plot(label=True)

In [76]: aperture.plot(frame='sci', label=True)
```



6 Pysiaf usage for SOC operations and support

From the outset, pysiaf was designed to support the operations of the JWST Science Operations Center (SOC) by implementing a robust way to maintain the SIAF products (cf. Figure 2). Here we present the typical usage scenarios in this context, where the main purpose of pysiaf will be to realize SIAF updates.

6.1 Keeping pysiaf synchronized with the SOC PRD

It is mandatory to keep the default SIAF that pysiaf uses synchronized with the latest SOC PRD release. This relies on a manual process because there is no public interface to the SIAF courtesy copies prepared by the PRD system. Some aspects could be further automated, but the current process is:

- Upon receipt of the STScI-internal release email the courtesy copy of the xml files have to be ingested in a pysiaf feature branch
- Open and merge a pull request
- Make a new release of pysiaf

This process is detailed under the *Procedures* tab at https://innerspace.stsci.edu/display/INSTEL/JWST+SIAF.

6.2 SIAF XML generation

As described above, the generation scripts of pysiaf use the source data to generate the SIAF XML product that can be ingested in the SOC PRD. The general structure of a generation script is detailed below:

- First, all apertures are created on the basis on the aperture definition file and their basic parameters are set. An ApertureCollection is created.
- In a second step, aperture dependencies are handled and the geometric parameters of all apertures are set.
- The third step takes care of setting the DDC aperture attribute.
- The aperture collection is then written to an XML file in the SIAF format compliant with the IRCD.
- In a fourth step, automated checks and comparisons of the SIAF product are performed and the products are moved to the pre-delivery directory.

All these steps are implemented sequentially in the script and only one single execution is required.

6.3 Generic SIAF update

SIAF updates are usually triggered by the operational needs of the observatory or the instruments. At the JWST SOC level the implementation of those changes are tracked through Jira issues in the JSOCINT and JSOCOPS projects, but the description of those processes is beyond the scope of this document.
Detailed technical documentation on the SIAF update itself is available under the *Procedures* tab at https://innerspace.stsci.edu/display/INSTEL/JWST+SIAF, and we provide an overview below:

- A JWSTSIAF Jira issue is used to track a SIAF update request at a higher level.
- The requestor updates the respective source data in a feature branch on their fork and submits these changes against the siaf-updates branch (this is the name of pysiaf's develop branch) via a pull request on the pysiaf GitHub repository.
- The STScI Telescope Branch's SIAF team processes the pull request, e.g. verifies that the updated SIAF products are generated correctly and all internal verifications pass, and interacts with the requestor as needed.
- The updated SIAF products are made available via the JIRA ticket and approval by the requestor is solicited.
- Upon approval, the feature branch is merged with the siaf-updates branch.
- When all SIAF changes have been implemented in this way, a PRD Change Request is

prepared and submitted by the Telescope Branch.
- After a new SOC PRD with those changes has been released, a release branch of pysiaf is created for clean-up and ingestion of the new PRD data.
- A new release of pysiaf is made.

The most common expected SIAF updates during commissioning are SIAF alignment updates and SIAF distortion updates. These will be triggered with changes to the concerned alignment and/or distortion source-data files and worked with the process described above. We note that this process has been successfully exercised for real pre-launch SIAF updates and for simulated SIAF updates during JWST commissioning rehearsals.

7 Suggestions for improvement for future SIAF implementations

Telescope aperture information like the one carried by the SIAF is necessary for the science operations of space observatories. It became clear that one added feature to the SIAF content would lead to significant improvements in terms of facilitating all associated interfaces with operational systems.
This change consists in adding a metadata flag to every SIAF item, both at aperture and at aperture-parameter level. This metadata flag should encode the customers of the respective item. In the case of JWST, potential customers correspond to e.g. the OSS (Onboard Script Subsystem), PPS (Proposal Planning Subsystem), DMS sub-systems. When changes to the SIAF are made, these flags can be evaluated automatically by software like pysiaf to produce a report that lists the affected areas and subsystems.
The upstream equivalent of this `downstream flag` would be a flag that encodes the origin of an item, which can be a piece of code, a dataset, or an individual. Technically, both flags can be implemented similar to the binary pixel data quality flags of the JWST pipeline.

Second, the content of the SIAF product that is ingested in the configuration-controlled operational database should be limited to items that have at least one downstream customer. Parameters that are used solely for visual or other inspection can be packaged automatically in independent products by processes like the pysiaf generate script.

Both improvements above are enabled by the use of software like pysiaf.

8 Conclusions

We described the pysiaf software package, its design principles and structure, and gave examples of its general usage and application for operations. We have already demonstrated repeatedly that pysiaf can handle the operational needs for maintaining the JWST SIAF product during pre-launch, commissioning, and operations.

9 References

Cox, C., Lallo, M., 2017, Description and Use of the JWST Science Instrument Aperture File, Technical Report JWST-STScI-001550, STScI

Groebner, A., 2017, Project Reference Database Subsystem To Users Interface Requirements

and Control Document Volume III: S&OC Subsystems, JWST-STScI-000949, STScI
Law, D., 2017, MIRI SIAF Input, Revision B, Technical Report, JWST-STScI-004741, STScI

Proffitt, C. R., Alves de Oliveira, C., Cox, C., et al., 2017, The Pre-Flight SI Aperture File, Part 4: NIRSpec, Technical Report JWST-STScI-005921, STScI

Sahlmann, J., Nelan, E. P., Averbukh, J., et al., 2019a, A comprehensive approach to HST focal plane geometric calibration, ISR TEL 2019-1, STScI

Sahlmann, J., Osborne, S., Cox, C., et al., 2019b, spacetelescope/pysiaf, v0.6.1, https://doi.org/10.5281/zenodo.3517245