pysatNASA Documentation

Release 0.0.4-alpha

Klenzing, Jeff, Stoneback, Russell, Burrell, Angeline G., Smith, Jo

Nov 07, 2022

CONTENTS

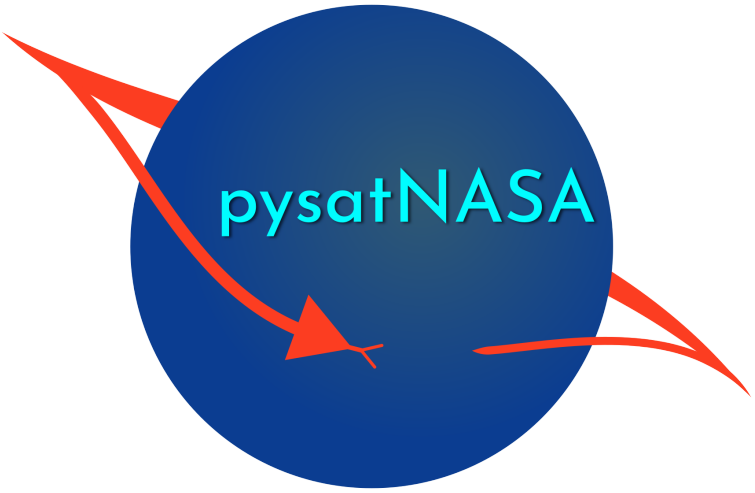
This documentation describes the pysatNASA module, which contains routines to NASA space science data as pysat.Instrument objects through the CDAWeb interface.

CHAPTER

ONE

OVERVIEW

This is a library of `pysat` instrument modules and methods designed to support NASA instruments and missions archived at the Community Data Analysis Web portal.



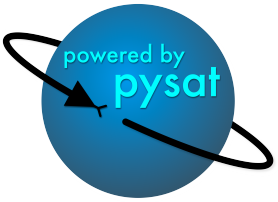
CHAPTER

TWO

INSTALLATION

The following instructions will allow you to install pysatNASA.

2.1 Prerequisites



pysatNASA uses common Python modules, as well as modules developed by and for the Space Physics community. This module officially supports Python 3.8+ and pysat 3.0.4+.

Common modules	Community modules
beautifulsoup4	cdflib>=0.4.4
lxml	pysat>=3.0.4
netCDF4	
numpy	
pandas	
requests	
xarray<2022.11	

2.2 Installation Options

1. Clone the git repository

```
git clone https://github.com/pysat/pysatNASA.git
```

2. Install pysatNASA: Change directories into the repository folder and run the setup.py file. There are a few ways you can do this:

   A. Install on the system (root privileges required):

   ```
   sudo python setup.py install
   ```

   B. Install at the user level:

```
python setup.py install --user
```

C. Install with the intent to develop locally:

```
python setup.py develop --user
```

Attention: This module has the following additional requirement:

```
pysatCDF
```

This can be installed as follows:

```
$ python -m pip install pysatNASA[all]
```

2.3 Post Installation

After installation, you may register the `pysatNASA` `Instrument` sub-modules with pysat. If this is your first time using pysat, check out the quickstart guide for pysat. Once pysat is set up, you may choose to register the the `pysatNASA` `Instruments` sub-modules by:

```
import pysat
import pysatNASA

pysat.utils.registry.register_by_module(pysatNASA.instruments)
```

You may then use the pysat `platform` and `name` keywords to initialize the model `Instrument` instead of the `inst_module` keyword argument.

CHAPTER

THREE

CITATION GUIDELINES

When publishing work that uses pysatNASA, please cite the package and any package it depends on that plays an important role in your analysis. Specifying which version of pysatNASA used will also improve the reproducibility of your presented results.

3.1 pysatNASA

The most recent citation can be found at Zenodo. The examples here are from the first release.

- Klenzing, J., et al. (2020). pysat/pysatNASA: Initial Release (Version 0.0.1). Zenodo. doi:10.5281/zenodo.3986132

```
@software{pysatNASA,
  author       = {Klenzing, J. and
                  Stoneback, R.A. and
                  Burrell, A.G. and
                  Pembroke, A. and
                  Depew, M. and
                  Spence, C.},
  title        = {pysat/pysatNASA: Initial Release},
  month        = aug,
  year         = 2020,
  publisher    = {Zenodo},
  version      = {v0.0.1},
  doi          = {10.5281/zenodo.3986132},
  url          = {https://doi.org/10.5281/zenodo.3986132}
}
```

CHAPTER

FOUR

SUPPORTED INSTRUMENTS

4.1 C/NOFS IVM

Module for the C/NOFS IVM instrument.

Supports the Ion Velocity Meter (IVM) onboard the Communication and Navigation Outage Forecasting System (C/NOFS) satellite, part of the Coupled Ion Netural Dynamics Investigation (CINDI). Downloads data from the NASA Coordinated Data Analysis Web (CDAWeb) in CDF format.

The IVM is composed of the Retarding Potential Analyzer (RPA) and Drift Meter (DM). The RPA measures the energy of plasma along the direction of satellite motion. By fitting these measurements to a theoretical description of plasma the number density, plasma composition, plasma temperature, and plasma motion may be determined. The DM directly measures the arrival angle of plasma. Using the reported motion of the satellite the angle is converted into ion motion along two orthogonal directions, perpendicular to the satellite track.

4.1.1 References

A brief discussion of the C/NOFS mission and instruments can be found at de La Beaujardière, O., et al. (2004), C/NOFS: A mission to forecast scintillations, J. Atmos. Sol. Terr. Phys., 66, 1573–1591, doi:10.1016/j.jastp.2004.07.030.

Discussion of cleaning parameters for ion drifts can be found in: Burrell, Angeline G., Equatorial topside magnetic field-aligned ion drifts at solar minimum, The University of Texas at Dallas, ProQuest Dissertations Publishing, 2012. 3507604.

Discussion of cleaning parameters for ion temperature can be found in: Hairston, M. R., W. R. Coley, and R. A. Heelis (2010), Mapping the duskside topside ionosphere with CINDI and DMSP, J. Geophys. Res.,115, A08324, doi:10.1029/2009JA015051.

4.1.2 Properties

platform
'cnofs'

name
'ivm'

tag
None supported

inst_id
None supported

4.1.3 Warnings

- The sampling rate of the instrument changes on July 29th, 2010. The rate is attached to the instrument object as .sample_rate.
- The cleaning parameters for the instrument are still under development.

pysatNASA.instruments.cnofs_ivm.**clean**(*self*)

Clean C/NOFS IVM data to the specified level.

pysatNASA.instruments.cnofs_ivm.**preprocess**(*self*)

Apply C/NOFS IVM default attributes.

4.2 C/NOFS PLP

Module for the C/NOFS PLP instrument.

Supports the Planar Langmuir Probe (PLP) onboard the Communication and Navigation Outage Forecasting System (C/NOFS) satellite. Downloads data from the NASA Coordinated Data Analysis Web (CDAWeb).

Description from CDAWeb:

The Planar Langmuir Probe on C/NOFS is a suite of 2 current measuring sensors mounted on the ram facing surface of the spacecraft. The primary sensor is an Ion Trap (conceptually similar to RPAs flown on many other spacecraft) capable of measuring ion densities as low as 1 cm-3 with a 12 bit log electrometer. The secondary senor is a swept bias planar Langmuir probe (Surface Probe) capable of measuring Ne, Te, and spacecraft potential.

The ion number density is the one second average of the ion density sampled at either 32, 256, 512, or 1024 Hz (depending on the mode).

The ion density standard deviation is the standard deviation of the samples used to produce the one second average number density.

DeltaN/N is the detrened ion number density 1 second standard deviation divided by the mean 1 sec density.

The electron density, electron temperature, and spacecraft potential are all derived from a least squares fit to the current-bias curve from the Surface Probe.

The data is PRELIMINARY, and as such, is intended for BROWSE PURPOSES ONLY.

4.2.1 References

A brief discussion of the C/NOFS mission and instruments can be found at de La Beaujardière, O., et al. (2004), C/NOFS: A mission to forecast scintillations, J. Atmos. Sol. Terr. Phys., 66, 1573–1591, doi:10.1016/j.jastp.2004.07.030.

4.2.2 Properties

platform
'cnofs'

name
'plp'

tag
None supported

inst_id
None supported

4.2.3 Warnings

- The data are PRELIMINARY, and as such, are intended for BROWSE PURPOSES ONLY.
- Currently no cleaning routine.
- Module not written by PLP team.

pysatNASA.instruments.cnofs_plp.**clean**(*self*)
Clean C/NOFS PLP data to the specified level.

4.3 C/NOFS VEFI

Module for the C/NOFS VEFI instrument.

Supports the Vector Electric Field Instrument (VEFI) onboard the Communication and Navigation Outage Forecasting System (C/NOFS) satellite. Downloads data from the NASA Coordinated Data Analysis Web (CDAWeb).

Description from CDAWeb (with updated link):

The DC vector magnetometer on the CNOFS spacecraft is a three axis, fluxgate sensor with active thermal control situated on a 0.6m boom. This magnetometer measures the Earth's magnetic field using 16 bit A/D converters at 1 sample per sec with a range of .. 45,000 nT. Its primary objective on the CNOFS spacecraft is to enable an accurate V x B measurement along the spacecraft trajectory. In order to provide an in-flight calibration of the magnetic field data, we compare the most recent POMME model (the POtsdam Magnetic Model of the Earth, https://geomag.us/models/pomme5.html) with the actual magnetometer measurements to help determine a set of calibration parameters for the gains, offsets, and non-orthogonality matrix of the sensor axes. The calibrated magnetic field measurements are provided in the data file here. The VEFI magnetic field data file currently contains the following variables: B_north Magnetic field in the north direction B_up Magnetic field in the up direction B_west Magnetic field in the west direction

The data is PRELIMINARY, and as such, is intended for BROWSE PURPOSES ONLY.

4.3.1 References

A brief discussion of the C/NOFS mission and instruments can be found at de La Beaujardière, O., et al. (2004), C/NOFS: A mission to forecast scintillations, J. Atmos. Sol. Terr. Phys., 66, 1573–1591, doi:10.1016/j.jastp.2004.07.030.

4.3.2 Properties

platform
'cnofs'

name
'vefi'

tag
Select measurement type, one of {'dc_b'}

inst_id
None supported

4.3.3 Note

- tag = 'dc_b': 1 second DC magnetometer data

4.3.4 Warnings

- The data is PRELIMINARY, and as such, is intended for BROWSE PURPOSES ONLY.
- Limited cleaning routine.
- Module not written by VEFI team.

pysatNASA.instruments.cnofs_vefi.**clean**(*self*)
Clean VEFI data to the specified level.

4.4 DE2 LANG

Module for the DE2 LANG instrument.

Supports the Langmuir Probe (LANG) instrument on Dynamics Explorer 2 (DE2).

From CDAWeb:

The Langmuir Probe Instrument (LANG) was a cylindrical electrostatic probe that obtained measurements of electron temperature, Te, and electron or ion concentration, Ne or Ni, respectively, and spacecraft potential. Data from this investigation were used to provide temperature and density measurements along magnetic field lines related to thermal energy and particle flows within the magnetosphere-ionosphere system, to provide thermal plasma conditions for wave-particle interactions, and to measure large-scale and fine-structure ionospheric effects of energy deposition in the ionosphere. The Langmuir Probe instrument was identical to that used on the AE satellites and the Pioneer Venus Orbiter. Two independent sensors were connected to individual adaptive sweep voltage circuits which continuously tracked the changing electron temperature and spacecraft potential, while autoranging electrometers adjusted their gain in response to the changing plasma density. The control signals used to achieve this automatic tracking provided a continuous monitor of the ionospheric parameters without telemetering each volt-ampere (V-I) curve. Furthermore,

internal data storage circuits permitted high resolution, high data rate sampling of selected V-I curves for transmission to ground to verify or correct the inflight processed data. Time resolution was 0.5 seconds.

4.4.1 References

J. P. Krehbiel, L. H. Brace, R. F. Theis, W. H. Pinkus, and R. B. Kaplan, "The Dynamics Explorer 2 Langmuir Probe (LANG)", Space Sci. Instrum., 5, 493-502, 1981.

4.4.2 Properties

platform
'de2'

name
'lang'

inst_id
None Supported

tag
None Supported

4.4.3 Warnings

- Currently no cleaning routine.

4.5 DE2 NACS

The DE2 NACS instrument.

Supports the Neutral Atmosphere Composition Spectrometer (NACS) instrument on Dynamics Explorer 2 (DE2).

From CDAWeb:

The Neutral Atmosphere Composition Spectrometer (NACS) was designed to obtain in situ measurements of the neutral atmospheric composition and to study the variations of the neutral atmosphere in response to energy coupled into it from the magnetosphere. Because temperature enhancements, large-scale circulation cells, and wave propagation are produced by energy input (each of which posseses a specific signature in composition variation), the measurements permitted the study of the partition, flow, and deposition of energy from the magnetosphere. Specifically, the investigation objective was to characterize the composition of the neutral atmosphere with particular emphasis on variability in constituent densities driven by interactions in the atmosphere, ionosphere, and magnetosphere system. The quadrupole mass spectrometer used was nearly identical to those flown on the AE-C, -D, and -E missions. The electron- impact ion source was used in a closed mode. Atmospheric particles entered an antechamber through a knife-edged orifice, where they were thermalized to the instrument temperature. The ions with the selected charge-to-mass ratios had stable trajectories through the hyperbolic electric field, exited the analyzer, and entered the detection system. An off-axis beryllium-copper dynode multiplier operating at a gain of 2.E6 provided an output pulse of electrons for each ion arrival. The detector output had a pulse rate proportional to the neutral density in the ion source of the selected mass. The instrument also included two baffles that scanned across the input orifice for optional measurement of the zonal and vertical components of the neutral wind. The mass select system provided for 256 mass values between 0 and 51 atomic mass units (u) or each 0.2 u. It was possible to call any one of these mass numbers into each of eight 0.016-s intervals. This sequence was repeated each 0.128 s.

This data set includes daily files of the PI-provided DE-2 NACS 1-second data and corresponding orbit parameters. The data set was generated at NSSDC from the original PI-provided data and software (SPTH-00010) and from the orbit/attitude database and software that is part of the DE-2 UA data set (SPIO-00174). The original NACS data were provided by the PI team in a highly compressed VAX/VMS binary format on magnetic tapes. The data set covers the whole DE-2 mission time period. Each data point is an average over the normally 8 measurements per second. Densities and relative errors are provided for atomic oxygen (O), molecular nitrogen (N2), helium (He), atomic nitrogen (N), and argon (Ar). The data quality is generally quite good below 500 km, but deteriorates towards higher altitudes as oxygen and molecular nitrogen approach their background values (which could only be determined from infrequent spinning orbits) and the count rate for Ar becomes very low. The difference between minimum (background) and maximum count rate for atomic nitrogen (estimated from mass 30) was so small that results are generally poor. Data were lost between 12 March 1982 and 31 March 1982 when the counter overflowed.

4.5.1 References

G. R. Carrignan, B. P. Block, J. C. Maurer, A. E. Hedin, C. A. Reber, N. W. Spencer, "The neutral mass spectrometer on Dynamics Explorer B", Space Sci. Instrum., 5, 429-441, 1981.

4.5.2 Properties

platform
'de2'

name
'nacs'

inst_id
None Supported

tag
None Supported

4.5.3 Warnings

- Currently no cleaning routine.

4.6 DE2 RPA

Module for the DE2 RPA instrument.

Supports the Retarding Potential Analyzer (RPA) instrument on Dynamics Explorer 2 (DE2).

From CDAWeb:

The Retarding Potential Analyzer (RPA) measured the bulk ion velocity in the direction of the spacecraft motion, the constituent ion concentrations, and the ion temperature along the satellite path. These parameters were derived from a least squares fit to the ion number flux vs energy curve obtained by sweeping or stepping the voltage applied to the internal retarding grids of the RPA. In addition, a separate wide aperture sensor, a duct sensor, was flown to measure the spectral characteristics of iregularities in the total ion concentration. The measured parameters obtained from this investigation were important to the understanding of mechanisms that influence the plasma; i.e., to understand the coupling between the solar wind and the earth's atmosphere. The measurements were made with a multigridded planar retarding potential analyzer very similar in concept and geometry to the instruments carried on the AE satellites. The retarding potential was variable in the range from approximately +32 to 0 V. The details of this voltage trace, and whether it was continuous or stepped, depended on the operating mode of the instrument. Specific parameters deduced

from these measurements were ion temperature; vehicle potential; ram component of the ion drift velocity; the ion and electron concentration irregularity spectrum; and the concentration of H+, He+, O+, and Fe+, and of molecular ions near perigee.

It includes the DUCT portion of the high resolutiondata from the Dynamics Explorer 2 (DE-2) Retarding Potential Analyzer (RPA) for the whole DE-2 mission time period in ASCII format. This version was generated at NSSDC from the PI-provided binary data (SPIO-00232). The DUCT files include RPA measurements ofthe total ion concentration every 64 times per second. Due to a failure in the instrument memory system RPA data are not available from 81317 06:26:40 UT to 82057 13:16:00 UT. This data set is based on the revised version of the RPA files that was submitted by the PI team in June of 1995. The revised RPA data include a correction to the spacecraft potential.

4.6.1 References

W. B. Hanson, R. A. Heelis, R. A. Power, C. R. Lippincott, D. R. Zuccaro, B. J. Holt, L. H. Harmon, and S. Sanatani, "The retarding potential analyzer for dynamics explorer-B," Space Sci. Instrum. 5, 503–510 (1981).

4.6.2 Properties

platform
'de2'

name
'rpa'

inst_id
None Supported

tag
None Supported

4.6.3 Warnings

- Currently no cleaning routine.

4.7 DE2 WATS

Module for the DE2 WATS instrument.

Supports the Wind and Temperature Spectrometer (WATS) instrument on Dynamics Explorer 2 (DE2).

From CDAWeb:

The Wind and Temperature Spectrometer (WATS) measured the in situ neutral winds, the neutral particle temperatures, and the concentrations of selected gases. The objective of this investigation was to study the interrelationships among the winds, temperatures, plasma drift, electric fields, and other properties of the thermosphere that were measured by this and other instruments on the spacecraft. Knowledge of how these properties are interrelated contributed to an understanding of the consequences of the acceleration of neutral particles by the ions in the ionosphere, the acceleration of ions by neutrals creating electric fields, and the related energy transfer between the ionosphere and the magnetosphere. Three components of the wind, one normal to the satellite velocity vector in the horizontal plane, one vertical, and one in the satellite direction were measured. A retarding potential quadrupole mass spectrometer, coupled to the atmosphere through a precisely orificed antechamber, was used. It was operated in either of two modes: one employed the retarding capability and the other used the ion source as a conventional nonretarding source. Two scanning baffles were used in front of the mass spectrometer: one moved vertically and the other moved horizontally. The magnitudes

of the horizontal and vertical components of the wind normal to the spacecraft velocity vector were computed from measurements of the angular relationship between the neutral particle stream and the sensor. The component of the total stream velocity in the satellite direction was measured directly by the spectrometer system through determination of the required retarding potential. At altitudes too high for neutral species measurements, the planned operation required the instrument to measure the thermal ion species only. A series of four sequentially occurring slots –each a 2-s long measurement interval– was adapted for the basic measurement format of the instrument. Different functions were commanded into these slots in any combination, one per measurement interval. Thus the time resolution can be 2, 4, 6, or 8 seconds. Further details are found in This data set consists of the high-resolution data of the Dynamics Explorer 2 Wind and Temperature Spectrometer (WATS) experiment. The files contain the neutral density, temperature and horizontal (zonal) wind velocity, and orbital parameters in ASCII format. The time resolution is typically 2 seconds. Data are given as daily files (typically a few 100 Kbytes each). PI-provided software (WATSCOR) was used to correct the binary data set. NSSDC-developed software was used to add the orbit parameters, to convert the binary into ASCII format and to combine the (PI-provided) orbital files into daily files. For more on DE-2, WATS, and the binary data, see the WATS_VOLDESC_SFDU_DE.DOC and WATS_FORMAT_SFDU_DE.DOC files. More information about the processing done at NSSDC is given in WATS_NSSDC_PRO_DE.DOC.

4.7.1 References

N. W. Spencer, L. E. Wharton, H. B. Niemann, A. E. Hedin, G. R. Carrignan, J. C. Maurer, “The Dynamics Explorer Wind and Temperature Spectrometer”, Space Sci. Instrum., 5, 417-428, 1981.

4.7.2 Properties

platform
‘de2’

name
‘wats’

inst_id
None Supported

tag
None Supported

4.7.3 Warnings

- Currently no cleaning routine.

4.8 FORMOSAT-1 IVM

Module for the ICON EUV instrument.

Supports the Ion Velocity Meter (IVM) onboard the Formosat-1 (formerly ROCSAT-1) mission. Downloads data from the NASA Coordinated Data Analysis Web (CDAWeb).

4.8.1 Properties

platform
'formosat1'

name
'ivm'

tag
None

inst_id
None supported

4.8.2 Warnings

- Currently no cleaning routine.

pysatNASA.instruments.formosat1_ivm.**init**(*self*)

Initialize the Instrument object with instrument specific values.

Runs once upon instantiation.

4.9 ICON EUV

Module for the ICON EUV instrument.

Supports the Extreme Ultraviolet (EUV) imager onboard the Ionospheric CONnection Explorer (ICON) satellite. Accesses local data in netCDF format.

4.9.1 Properties

platform
'icon'

name
'euv'

tag
None supported

4.9.2 Warnings

- The cleaning parameters for the instrument are still under development.
- Only supports level-2 data.

4.9.3 Examples

```
import pysat
euv = pysat.Instrument(platform='icon', name='euv')
euv.download(dt.datetime(2020, 1, 1), dt.datetime(2020, 1, 31))
euv.load(2020, 1)
```

By default, pysat removes the ICON level tags from variable names, ie, ICON_L27_Ion_Density becomes Ion_Density. To retain the original names, use

```
euv = pysat.Instrument(platform='icon', name='euv',
                       keep_original_names=True)
```

pysatNASA.instruments.icon_euv.**clean**(*self*)

Clean ICON EUV data to the specified level.

pysatNASA.instruments.icon_euv.**filter_metadata**(*meta_dict*)

Filter EUV metadata to remove warnings during loading.

Parameters

meta_dict
[dict] Dictionary of metadata from file

Returns

meta_dict
[dict] Filtered EUV metadata

pysatNASA.instruments.icon_euv.**load**(*fnames*, *tag=''*, *inst_id=''*, *keep_original_names=False*)

Load ICON EUV data into *xarray.Dataset* object and *pysat.Meta* objects.

This routine is called as needed by pysat. It is not intended for direct user interaction.

Parameters

fnames
[array-like] Iterable of filename strings, full path, to data files to be loaded. This input is nominally provided by pysat itself.

tag
[str] Tag name used to identify particular data set to be loaded. This input is nominally provided by pysat itself. (default='')

inst_id
[str] Instrument ID used to identify particular data set to be loaded. This input is nominally provided by pysat itself. (default='')

keep_original_names
[bool] If True then the names as given in the netCDF ICON file will be used as is. If False, a preamble is removed. (default=False)

Returns

data
[xr.Dataset] An xarray Dataset with data prepared for the pysat.Instrument

meta
[pysat.Meta] Metadata formatted for a pysat.Instrument object.

Examples

```
inst = pysat.Instrument('icon', 'euv', tag='', inst_id='a')
inst.load(2020, 1)
```

pysatNASA.instruments.icon_euv.**preprocess**(*self*, *keep_original_names=False*)

Adjust epoch timestamps to datetimes and remove variable preambles.

Parameters

keep_original_names
[bool] if True then the names as given in the netCDF ICON file will be used as is. If False, a preamble is removed. (default=False)

4.10 ICON FUV

Module for the ICON FUV instrument.

Supports the Far Ultraviolet (FUV) imager onboard the Ionospheric CONnection Explorer (ICON) satellite. Accesses local data in netCDF format.

4.10.1 Properties

platform
'icon'

name
'fuv'

tag
None supported

4.10.2 Warnings

- The cleaning parameters for the instrument are still under development.
- Only supports level-2 data.

4.10.3 Example

```
import pysat
fuv = pysat.Instrument(platform='icon', name='fuv', tag='day')
fuv.download(dt.datetime(2020, 1, 1), dt.datetime(2020, 1, 31))
fuv.load(2020, 1)
```

By default, pysat removes the ICON level tags from variable names, ie, ICON_L27_Ion_Density becomes Ion_Density. To retain the original names, use

```
fuv = pysat.Instrument(platform='icon', name='fuv', tag=day',
                       keep_original_names=True)
```

pysatNASA.instruments.icon_fuv.**clean**(*self*)

Clean ICON FUV data to the specified level.

pysatNASA.instruments.icon_fuv.**filter_metadata**(*meta_dict*)

Filter FUV metadata to remove warnings during loading.

Parameters

meta_dict
[dict] Dictionary of metadata from file

Returns

meta_dict
[dict] Filtered FUV metadata

pysatNASA.instruments.icon_fuv.**load**(*fnames*, *tag=''*, *inst_id=''*, *keep_original_names=False*)

Load ICON FUV data into xarray.Dataset object and pysat.Meta objects.

This routine is called as needed by pysat. It is not intended for direct user interaction.

Parameters

fnames
[array-like] iterable of filename strings, full path, to data files to be loaded. This input is nominally provided by pysat itself.

tag
[str] Tag name used to identify particular data set to be loaded. This input is nominally provided by pysat itself. (default='')

inst_id
[str] Instrument ID used to identify particular data set to be loaded. This input is nominally provided by pysat itself. (default='')

keep_original_names
[bool] if True then the names as given in the netCDF ICON file will be used as is. If False, a preamble is removed. (default=False)

Returns

data
[xr.Dataset] An xarray Dataset with data prepared for the pysat.Instrument

meta
[pysat.Meta] Metadata formatted for a pysat.Instrument object.

Examples

```
inst = pysat.Instrument('icon', 'fuv')
inst.load(2020, 1)
```

pysatNASA.instruments.icon_fuv.**preprocess**(*self*, *keep_original_names=False*)

Adjust epoch timestamps to datetimes and remove variable preambles.

Parameters

keep_original_names
[bool] if True then the names as given in the netCDF ICON file will be used as is. If False, a preamble is removed. (default=False)

4.11 ICON IVM

Module for the ICON IVM instrument.

Supports the Ion Velocity Meter (IVM) onboard the Ionospheric Connections (ICON) Explorer.

4.11.1 Properties

platform
'icon'

name
'ivm'

tag
None supported

inst_id
'a' or 'b'

4.11.2 Example

```
import pysat
ivm = pysat.Instrument(platform='icon', name='ivm', inst_id='a')
ivm.download(dt.datetime(2020, 1, 1), dt.datetime(2020, 1, 31))
ivm.load(2020, 1)
```

By default, pysat removes the ICON level tags from variable names, ie, ICON_L27_Ion_Density becomes Ion_Density. To retain the original names, use

```
ivm = pysat.Instrument(platform='icon', name='ivm', inst_id='a',
                       keep_original_names=True)
```

4.11.3 Author

R. A. Stoneback

pysatNASA.instruments.icon_ivm.**clean**(*self*)

Clean ICON IVM data to the specified level.

pysatNASA.instruments.icon_ivm.**filter_metadata**(*meta_dict*)

Filter IVM metadata to remove warnings during loading.

Parameters

meta_dict
[dict] Dictionary of metadata from file

Returns

dict
Filtered IVM metadata

pysatNASA.instruments.icon_ivm.**load**(*fnames*, *tag=''*, *inst_id=''*, *keep_original_names=False*)

Load ICON IVM data into *pandas.DataFrame* and *pysat.Meta* objects.

This routine is called as needed by pysat. It is not intended for direct user interaction.

Parameters

fnames
[array-like] iterable of filename strings, full path, to data files to be loaded. This input is nominally provided by pysat itself.

tag
[str] Tag name used to identify particular data set to be loaded. This input is nominally provided by pysat itself. (default='')

inst_id
[str] Instrument ID used to identify particular data set to be loaded. This input is nominally provided by pysat itself. (default='')

keep_original_names
[bool] if True then the names as given in the netCDF ICON file will be used as is. If False, a preamble is removed. (default=False)

Returns

data
[pds.DataFrame] A pandas DataFrame with data prepared for the pysat.Instrument

meta
[pysat.Meta] Metadata formatted for a pysat.Instrument object.

Examples

```
inst = pysat.Instrument('icon', 'ivm', inst_id='a', tag='')
inst.load(2020, 1)
```

pysatNASA.instruments.icon_ivm.**preprocess**(*self*, *keep_original_names=False*)

Remove variable preambles.

Parameters

keep_original_names
[bool] if True then the names as given in the netCDF ICON file will be used as is. If False, a preamble is removed. (default=False)

4.12 ICON MIGHTI

Module for the ICON MIGHTI instrument.

Supports the Michelson Interferometer for Global High-resolution Thermospheric Imaging (MIGHTI) instrument onboard the Ionospheric CONnection Explorer (ICON) satellite. Accesses local data in netCDF format.

4.12.1 Properties

platform
'icon'

name
'mighti'

tag
Supports 'los_wind_green', 'los_wind_red', 'vector_wind_green', 'vector_wind_red', 'temperature'. Note that not every data product available for every inst_id

inst_id
'vector', 'a', or 'b'

4.12.2 Warnings

- The cleaning parameters for the instrument are still under development.
- Only supports level-2 data.

4.12.3 Example

```
import pysat
mighti = pysat.Instrument('icon', 'mighti', tag='vector_wind_green',
                          inst_id='vector', clean_level='clean')
mighti.download(dt.datetime(2020, 1, 30), dt.datetime(2020, 1, 31))
mighti.load(2020, 2)
```

By default, pysat removes the ICON level tags from variable names, ie, ICON_L27_Ion_Density becomes Ion_Density. To retain the original names, use

```
mighti = pysat.Instrument(platform='icon', name='mighti',
                          tag='vector_wind_green', inst_id='vector',
                          clean_level='clean',
                          keep_original_names=True)
```

4.12.4 Note

Currently red and green data products are bundled together in zip files on the server. This results in 'double downloading'. This will be fixed once data is transfered to SPDF.

pysatNASA.instruments.icon_mighti.**clean**(*self*)
Clean ICON MIGHTI data to the specified level.

pysatNASA.instruments.icon_mighti.**filter_metadata**(*meta_dict*)
Filter FUV metadata to remove warnings during loading.

Parameters

meta_dict
[dict] Dictionary of metadata from file

Returns

meta_dict
[dict] Filtered FUV metadata

pysatNASA.instruments.icon_mighti.**load**(*fnames*, *tag=''*, *inst_id=''*, *keep_original_names=False*)

Load ICON MIGHTI data into *xarray.Dataset* and *pysat.Meta* objects.

This routine is called as needed by pysat. It is not intended for direct user interaction.

Parameters

fnames
[array-like] iterable of filename strings, full path, to data files to be loaded. This input is nominally provided by pysat itself.

tag
[str] Tag name used to identify particular data set to be loaded. This input is nominally provided by pysat itself. (default='')

inst_id
[str] Instrument ID used to identify particular data set to be loaded. This input is nominally provided by pysat itself. (default='')

keep_original_names
[bool] if True then the names as given in the netCDF ICON file will be used as is. If False, a preamble is removed. (default=False)

Returns

data
[xr.Dataset] An xarray Dataset with data prepared for the pysat.Instrument

meta
[pysat.Meta] Metadata formatted for a pysat.Instrument object.

Examples

```
inst = pysat.Instrument('icon', 'fuv')
inst.load(2020, 1)
```

pysatNASA.instruments.icon_mighti.**preprocess**(*self*, *keep_original_names=False*)

Adjust epoch timestamps to datetimes and remove variable preambles.

Parameters

keep_original_names
[bool] if True then the names as given in the netCDF ICON file will be used as is. If False, a preamble is removed. (default=False)

4.13 ISS FPMU

Module for the ISS FPMU instrument.

Supports the Floating Potential Measurement Unit (FPMU) instrument onboard the International Space Station (ISS). Downloads data from the NASA Coordinated Data Analysis Web (CDAWeb).

4.13.1 Properties

platform
'iss'

name
'fpmu'

tag
None Supported

inst_id
None supported

4.13.2 Warnings

- Currently clean only replaces fill values with Nans.
- Module not written by FPMU team.

pysatNASA.instruments.iss_fpmu.**clean**(*self*)
Clean ISS FPMU data to the specified level.

pysatNASA.instruments.iss_fpmu.**init**(*self*)
Initialize the Instrument object with instrument specific values.

Runs once upon instantiation.

4.14 JPL GPS

Module for the JPL GPS data products.

Supports ROTI data produced at JPL from International GNSS Service Total Electron Content (TEC)

The rate of TEC index (ROTI) characterizes TEC fluctuations observed along receiver-to-satellite line of sight links over a 5-minute interval. The measurement is obtained by processing GNSS dual-frequency phase data and computing the standard deviation of the rate of TEC change over that interval after removing its background variation trend.

ROTI data are provided as global maps using a 2.5 x 5 degree (geographic latitude x longitude) grid. The median ROTI value is calculated in each bin. GNSS data contributing to the ROTI computation are primarily collected from the global network of International GNSS Service and the regional network of Continuous Operating Reference Station (CORS).

4.14.1 References

Pi, X., A. J. Mannucci, U. J. Lindqwister, and C. M. Ho, Monitoring of global ionospheric irregularities using the worldwide GPS network, Geophys. Res. Lett., 24, 2283, 1997.

Pi, X., F. J. Meyer, K. Chotoo, Anthony Freeman, R. G. Caton, and C. T. Bridgwood, Impact of ionospheric scintillation on Spaceborne SAR observations studied using GNSS, Proc. ION-GNSS, pp.1998-2006, 2012.

4.14.2 Properties

platform
'jpl'

name
'gps'

tag
['roti']

inst_id
None supported

4.14.3 Warnings

- The cleaning parameters for the instrument are still under development.

`pysatNASA.instruments.jpl_gps.init(self)`

Initialize the Instrument object with instrument specific values.

Runs once upon instantiation.

4.15 OMNI HRO

Module for the OMNI HRO instrument.

Supports OMNI Combined, Definitive, IMF and Plasma Data, and Energetic Proton Fluxes, Time-Shifted to the Nose of the Earth's Bow Shock, plus Solar and Magnetic Indices. Downloads data from the NASA Coordinated Data Analysis Web (CDAWeb). Supports both 5 and 1 minute files.

4.15.1 Properties

platform
'omni'

name
'hro'

tag
Select time between samples, one of {'1min', '5min'}

inst_id
None supported

4.15.2 Note

Files are stored by the first day of each month. When downloading use omni.download(start, stop, freq='MS') to only download days that could possibly have data. 'MS' gives a monthly start frequency.

This material is based upon work supported by the National Science Foundation under Grant Number 1259508.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

4.15.3 Warnings

- Currently no cleaning routine. Though the CDAWEB description indicates that these level-2 products are expected to be ok.
- Module not written by OMNI team.

pysatNASA.instruments.omni_hro.**calculate_clock_angle**(**args*, ***kwargs*)

Wrap functions that use the decorator function.

pysatNASA.instruments.omni_hro.**calculate_imf_steadiness**(**args*, ***kwargs*)

Wrap functions that use the decorator function.

pysatNASA.instruments.omni_hro.**time_shift_to_magnetic_poles**(**args*, ***kwargs*)

Wrap functions that use the decorator function.

4.16 SES14 GOLD

Module for the SES14 GOLD instrument.

Supports the Nmax data product from the Global Observations of the Limb and Disk (GOLD) satellite. Accesses data in netCDF format.

4.16.1 Properties

platform
'ses14'

name
'gold'

tag
'nmax'

4.16.2 Warnings

- The cleaning parameters for the instrument are still under development.
- strict_time_flag must be set to False

4.16.3 Examples

```
import datetime as dt
import pysat
nmax = pysat.Instrument(platform='ses14', name='gold', tag='nmax'
                        strict_time_flag=False)
nmax.download(dt.datetime(2020, 1, 1), dt.datetime(2020, 1, 31))
nmax.load(2020, 1)
```

pysatNASA.instruments.ses14_gold.**init**(*self*)

Initialize the Instrument object with instrument specific values.

Runs once upon instantiation.

Parameters

self
[pysat.Instrument] Instrument class object

pysatNASA.instruments.ses14_gold.**load**(*fnames*, *tag=''*, *inst_id=''*)

Load GOLD NMAX data into *xarray.Dataset* and *pysat.Meta* objects.

This routine is called as needed by pysat. It is not intended for direct user interaction.

Parameters

fnames
[array-like] iterable of filename strings, full path, to data files to be loaded. This input is nominally provided by pysat itself.

tag
[str] Tag name used to identify particular data set to be loaded. This input is nominally provided by pysat itself. (default='')

inst_id
[str] Instrument ID used to identify particular data set to be loaded. This input is nominally provided by pysat itself. (default='')

****kwargs**
[extra keywords] Passthrough for additional keyword arguments specified when instantiating an Instrument object. These additional keywords are passed through to this routine by pysat.

Returns

data
[xr.Dataset] An xarray Dataset with data prepared for the pysat.Instrument

meta
[pysat.Meta] Metadata formatted for a pysat.Instrument object.

Examples

```
inst = pysat.Instrument('ses14', 'gold', tag='nmax')
inst.load(2019, 1)
```

4.17 TIMED SABER

The TIMED SABER instrument.

Supports the Sounding of the Atmosphere using Broadband Emission Radiometry (SABER) instrument on the Thermosphere Ionosphere Mesosphere Energetics Dynamics (TIMED) satellite.

4.17.1 Properties

platform
: [str] 'timed'

name
: [str] 'saber'

tag
: [str] None supported

inst_id
: [str] None supported

4.17.2 Note

Note on Temperature Errors: https://saber.gats-inc.com/temp_errors.php

SABER "Rules of the Road" for DATA USE Users of SABER data are asked to respect the following guidelines

- Mission scientific and model results are open to all.
- Guest investigators, and other members of the scientific community or general public should contact the PI or designated team member early in an analysis project to discuss the appropriate use of the data.
- Users that wish to publish the results derived from SABER data should normally offer co-authorship to the PI, Associate PI or designated team members. Co-authorship may be declined. Appropriate acknowledgement of institutions, personnel, and funding agencies should be given.
- Users should heed the caveats of SABER team members as to the interpretation and limitations of the data. SABER team members may insist that such caveats be published, even if co-authorship is declined. Data and model version numbers should also be specified.
- Pre-prints of publications and conference abstracts should be widely distributed to interested parties within the mission and related projects.

4.17.3 Warnings

- No cleaning routine

pysatNASA.instruments.timed_saber.**init**(*self*)

Initialize the Instrument object with instrument specific values.

Runs once upon instantiation.

4.18 TIMED SEE

Supports the SEE instrument on TIMED.

Downloads data from the NASA Coordinated Data Analysis Web (CDAWeb).

Supports two options for loading that may be specified at instantiation.

4.18.1 Properties

platform
'timed'

name
'see'

tag
None

inst_id
None supported

flatten_twod
If True, then two dimensional data is flattened across columns. Name mangling is used to group data, first column is 'name', last column is 'name_end'. In between numbers are appended 'name_1', 'name_2', etc. All data for a given 2D array may be accessed via, data.loc[:, 'item':'item_end'] If False, then 2D data is stored as a series of DataFrames, indexed by Epoch. data.loc[0, 'item'] (default=True)

4.18.2 Note

- no tag required

4.18.3 Warnings

- Currently no cleaning routine.

pysatNASA.instruments.timed_see.**init**(*self*)

Initialize the Instrument object with instrument specific values.

Runs once upon instantiation.

CHAPTER

FIVE

SUPPORTED CONSTELLATIONS

5.1 DE2

The Dynamics Explorer 2 spacecraft. Includes the instruments

- *DE2 LANG*
- *DE2 NACS*
- *DE2 RPA*
- *DE2 WATS*

5.2 ICON

The Ionosphere CONnection explorer spacecraft. Includes the instruments

- *ICON EUV*
- *ICON FUV*
- *ICON IVM*
- *ICON MIGHTI*

CHAPTER

SIX

EXAMPLES

Here are some examples that demonstrate how to use various pysatNASA tools

6.1 Loading ICON IVM data

pysatNASA uses pysat to load space science instrument data. As specified in the pysat tutorial, data may be loaded using the following commands. Data from the Ion Velocity Meter on board the Ionospheric CONnection Explorer (ICON) is used as an example.

```
import datetime as dt
import pysat
import pysatNASA as py_nasa

pysat.utils.registry.register_by_module(py_nasa.instruments)

old_time = dt.datetime(2020, 1, 1)
ivm = pysat.Instrument(platform='icon', name='ivm',
                       inst_id='a', update_files=True)
ivm.download(start=old_time)
ivm.load(date=old_time)
print(ivm)
```

The output shows some basic info about the instrument object, as well as information about the data loaded.

```
pysat Instrument object
-----------------------
Platform: 'icon'
Name: 'ivm'
Tag: ''
Instrument id: 'a'

Data Processing
---------------
Cleaning Level: 'clean'
Data Padding: None
Keyword Arguments Passed to list_files: {}
Keyword Arguments Passed to load: {}
Keyword Arguments Passed to preprocess: {}
Keyword Arguments Passed to download: {}
Keyword Arguments Passed to list_remote_files: {}
```

(continues on next page)

(continued from previous page)

```
Keyword Arguments Passed to clean: {}
Keyword Arguments Passed to init: {}
Custom Functions: 0 applied

Local File Statistics
---------------------
Number of files: 402
Date Range: 22 October 2019 --- 29 December 2020

Loaded Data Statistics
----------------------
Date: 01 January 2020
DOY: 001
Time range: 31 December 2019 23:59:57 --- 01 January 2020 23:59:55
Number of Times: 86400
Number of variables: 91

Variable Names:
A_Activity          A_Status            Altitude
                       ...
Unit_Vector_Zonal_X Unit_Vector_Zonal_Y Unit_Vector_Zonal_Z

pysat Meta object
-----------------
Tracking 21 metadata values
Metadata for 92 standard variables
Metadata for 0 ND variables
```

CHAPTER

SEVEN

GUIDE FOR DEVELOPERS

7.1 Contributor Covenant Code of Conduct

7.1.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

7.1.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

7.1.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

7.1.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

7.1.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at pysat.developers@gmail.com. The project team will review and investigate all complaints, and will respond in a way that it deems appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

7.1.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at https://contributor-covenant.org/version/1/4

7.2 Contributing

Bug reports, feature suggestions, and other contributions are greatly appreciated! pysat is a community-driven project and welcomes both feedback and contributions.

Come join us on Slack! An invitation to the pysat workspace is available in the 'About' section of the pysat GitHub Repository. Development meetings are generally held fortnightly.

7.2.1 Short version

- Submit bug reports and feature requests at GitHub
- Make pull requests to the `develop` branch

7.2.2 Bug reports

When reporting a bug please include:

- Your operating system name and version
- Any details about your local setup that might be helpful in troubleshooting
- Detailed steps to reproduce the bug

7.2.3 Feature requests and feedback

The best way to send feedback is to file an issue at GitHub.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that code contributions are welcome :)

7.2.4 Development

To set up `pysatNASA` for local development:

1. Fork pysatNASA on GitHub.
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/pysatNASA.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

   Tests for new instruments are performed automatically. See discussion here for more information on triggering these standard tests.

   Tests for custom functions should be added to the appropriately named file in `pysatNASA/tests`. For example, custom functions for the OMNI HRO data are tested in `pysatNASA/tests/test_omni_hro.py`. If no test file exists, then you should create one. This testing uses pytest, which will run tests on any python file in the test directory that starts with `test`. Classes must begin with `Test`, and methods must begin with `test` as well.

4. When you're done making changes, run all the checks to ensure that nothing is broken on your local system, as well as check for flake8 compliance:

```
pytest -vs --flake8 pysatNASA
```

5. Update/add documentation (in `docs`), if relevant
6. Add your name to the .zenodo.json file as an author
7. Commit your changes:

```
git add .
git commit -m "AAA: Brief description of your changes"
```

Where AAA is a standard shorthand for the type of change (eg, BUG or DOC). `pysat` follows the numpy development workflow, see the discussion there for a full list of this shorthand notation.

8. Once you are happy with the local changes, push to Github:

```
git push origin name-of-your-bugfix-or-feature
```

Note that each push will trigger the Continuous Integration workflow.

9. Submit a pull request through the GitHub website. Pull requests should be made to the `develop` branch.

7.2.5 Pull Request Guidelines

If you need some code review or feedback while you're developing the code, just make a pull request. Pull requests should be made to the `develop` branch.

For merging, you should:

1. Include an example for use
2. Add a note to `CHANGELOG.md` about the changes
3. Update the author list in `zenodo.json` if applicable
4. Ensure that all checks passed (current checks include Github Actions and Coveralls)

If you don't have all the necessary Python versions available locally or have trouble building all the testing environments, you can rely on GitHub Actions to run the tests for each change you add in the pull request. Because testing here will delay tests by other developers, please ensure that the code passes all tests on your local system first.

7.2.6 Project Style Guidelines

In general, pysat follows PEP8 and numpydoc guidelines. Pytest runs the unit and integration tests, flake8 checks for style, and sphinx-build performs documentation tests. However, there are certain additional style elements that have been adopted to ensure the project maintains a consistent coding style. These include:

- Line breaks should occur before a binary operator (ignoring flake8 W503)
- Combine long strings using `join`
- Preferably break long lines on open parentheses rather than using \
- Use no more than 80 characters per line
- Avoid using Instrument class key attribute names as unrelated variable names: `platform`, `name`, `tag`, and `inst_id`
- The pysat logger is imported into each sub-module and provides status updates at the info and warning levels (as appropriate)
- Several dependent packages have common nicknames, including:
  - `import datetime as dt`
  - `import numpy as np`
  - `import pandas as pds`
  - `import xarray as xr`
- When incrementing a timestamp, use `dt.timedelta` instead of `pds.DateOffset` when possible to reduce program runtime

- All classes should have `__repr__` and `__str__` functions
- Docstrings use `Note` instead of `Notes`
- Try to avoid creating a try/except statement where except passes
- Use setup and teardown in test classes
- Use pytest parametrize in test classes when appropriate
- Provide testing class methods with informative failure statements and descriptive, one-line docstrings
- Block and inline comments should use proper English grammar and punctuation with the exception of single sentences in a block, which may then omit the final period

CHAPTER

EIGHT

MIGRATION FROM PYSAT 2

With the release of pysat 3.0.0, the pysat project now keeps instrument modules within distinct packages. Each of these packages acts as an interface between the core pysat package and a unique data provider. pysatNASA fills this role for the Space Physics Data Facility at NASA.

8.1 Registering the pysatNASA library

While each module can be loaded separately, users may find it easier to register all instruments.

```
import pysat
import pysatNASA
pysat.utils.registry.register_by_module(pysatNASA.instruments)
```

This creates a shortcut so that instruments may be loaded using only `platform` and `name` without having to load the instrument package each time.

```
import pysat
ivm = pysat.Instrument('cnofs', 'ivm')
```

8.2 Modifying the directory structure

The internal directory structure has been updated in pysat 3.0.0 to include a separate layer for `inst_id`. Users who have already downloaded data in a previous version should follow this tutorial to make their local data directories compatible with the new version.

8.3 A Note about ICON data

Starting with pysatNASA 0.0.2, the data for the Ionospheric CONnection Explorer (ICON) is now accessed from the SPDF server directly rather than the University of California at Berkeley server. There is a slight update in the file names at the new location, which is not compatible with the previous versions of pysat. It is recommended that users download this data using the new software.

CHAPTER

NINE

CHANGE LOG

All notable changes to this project will be documented in this file. This project adheres to Semantic Versioning.

9.1 [0.0.4] - 2022-11-07

- Update instrument tests with new test class
- Support xarray datasets through cdflib
- Preferentially loads data into pandas using pysatCDF if installed
- Adds pysatCDF to optional requirements invoked via '[all]' option at installation
- New Instruments
  - JPL GPS ROTI
- Bug Fixes
  - Fixed a bug in metadata when loading GOLD Nmax data.
  - Fixed a bug in user feedback for `methods.cdaweb.download`
  - Fixed a bug in loading ICON IVM data (added multi_file_day = True)
  - Allow for array-like OMNI HRO meta data
  - Fixed date handling for OMNI HRO downloads
  - Updated filenames for TIMED SABER
- Maintenance
  - Reduce duplication of code in instrument modules
  - Include flake8 linting of docstrings and style in Github Actions
  - Move OMNI HRO custom functions to a methods module
  - Deprecate OMNI HRO custom functions in instrument module
  - Update GitHub actions to the latest versions
  - Added downstream test to test code with pysat RC
  - Remove deprecated `convert_timestamp_to_datetime` calls
  - Remove deprecated pandas syntax
  - Added version cap for xarray 2022.11
- Documentation

  - New logo added

9.2 [0.0.3] - 2022-05-18

- Include flake8 linting of docstrings and style in Github Actions
- Include Windows tests in Github Actions
- Bug Fixes
  - Expanded cleaning of ICON IVM ion drifts to more variables
  - Fixed a bug in loading ICON IVM data (added multi_file_day = True)
  - Fixed a bug where OMNI meta data float values are loaded as arrays
  - Fixed metadata type issues when loading ICON instrument data.
- Maintenance
  - Removed dummy vars after importing instruments and constellations
  - Updated NEP29 compliance in Github Actions
  - Limit versions of hacking for improved pip compliance
  - Update instrument template standards
  - Updated documentation style
  - Removed cap on cdflib

9.3 [0.0.2] - 2021-06-07

- Updated Instruments and routines to conform with changes made for pysat 3.0
- Added documentation
- Instrument Changes
  - Preliminary support added for SES-14 GOLD Nmax
  - Updated cleaning routines for C/NOFS IVM data
  - Migrated remote server for ICON instruments to SPDF from UCB
  - Renamed ROCSAT1 IVM as FORMOSAT1 IVM
  - Dropped support for SPORT IVM (unlaunched, moved to pysatIncubator)
- Implements GitHub Actions as primary CI test environment
- Improved PEP8 compliance
- Replaced pysatCDF with cdflib support
- Bug Fixes
  - `remote_file_list` error if start/stop dates unspecified
  - Improved download robustness

9.4 [0.0.1] - 2020-08-13

- Initial port of existing routines from pysat

CHAPTER

TEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

T