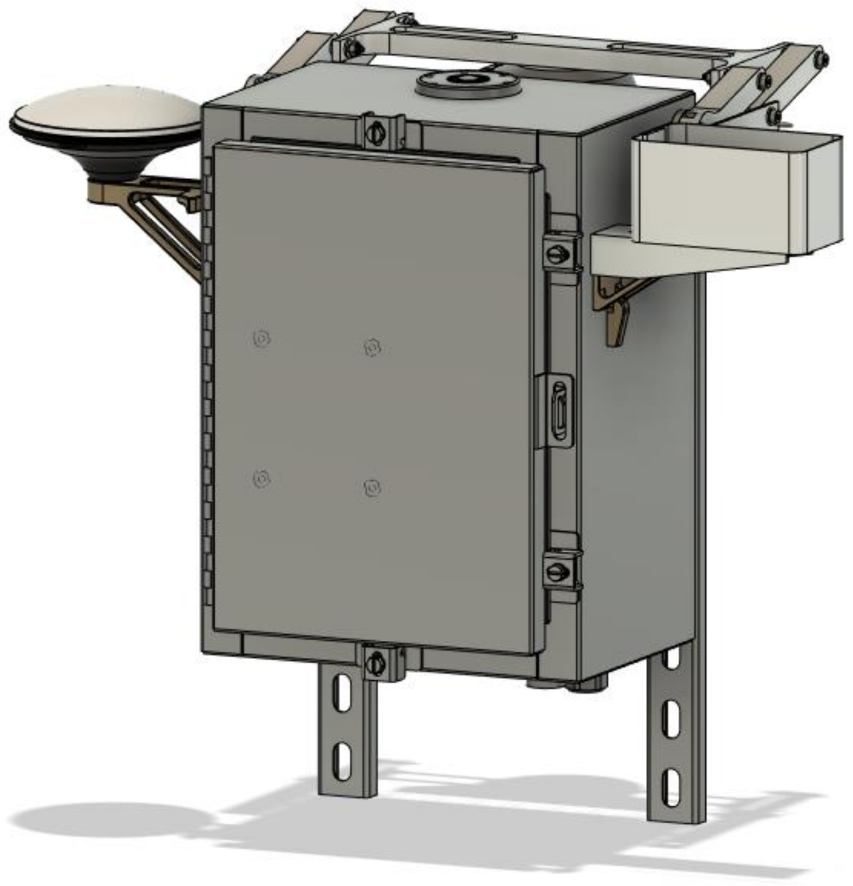
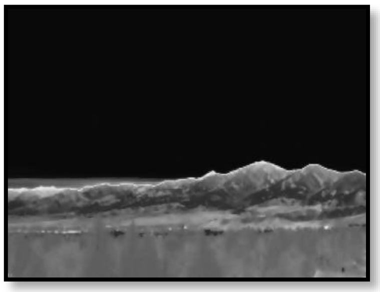
All-Sky Infrared Cloud Imager

Software Manual



NWB Sensors, Inc.

Contact Information



NWB Sensors, Inc.

Main Office	Mailing
NWB Sensors Inc 1716 W Main St Ste 8B Bozeman, MT 59715	NWB Sensors Inc 80555 Gallatin Road Bozeman, MT 59718
Support	**Sales**
info@nwbsensors.com	sales@nwbsensors.com
406.579.0510	406.579.0510

Table of Contents

Instrument Overview

The All-Sky Infrared Cloud Imager (ICI) detects and classifies clouds using a calibrated long-wave infrared (LWIR) thermal imaging camera. The system operates by observing the LWIR emission of the sky and clouds and then removing the emission of the sky to produce a residual radiance from which measured cloud products are derived.

The instrument is separated into two enclosures which are connected by cables. The Camera Enclosure mounts outdoors, and the Power Enclosure mounts inside a climate-controlled building. The exterior Camera Enclosure is a climate-controlled aluminum box that houses the camera and supporting electronics. The Camera Enclosure has a hatch that closes during rain or high winds to protect the camera lens. The Power Enclosure houses the power supplies and control circuitry.

Scope

This document provides a guide to operating the instrument using its provided software interfaces.

Supporting Documents

This document is one of several which describe the installation, maintenance, and operation of the ICI. As appropriate, refer to the related documents provided with the instrument:

- *All-Sky Infrared Cloud Imager Installation Manual*
- *All-Sky Infrared Cloud Imager Hardware Maintenance Manual*
- *All-Sky Infrared Cloud Imager Software Operations Manual*
- *All-Sky Infrared Cloud Imager Camera & Lens Swap Manual (optional)*

Instrument Fact Sheet

The following parameters are needed to communicate with the instrument but should be filled out as necessary by site personnel in accordance with site security procedures. A default host name and user password will be assigned by NWB Sensors prior to delivery. A specific host name can be preconfigured prior to shipment at the customer’s request.

Parameter	Value
Host name	
IP address	
SSH username	ici
SSH password	

Infrared Cloud Imager Software Design

Overview

The ICI software resides on a Jetson embedded computer within the Power Enclosure. This computer connects to the outside world over fiber or Ethernet through a switch in the Power Enclosure. Several USB devices in the Camera Enclosure connect to the embedded computer over a USB-over-fiber extender. These USB devices include the thermal camera, the Global Navigation Satellite System (GNSS) receiver and the weather sensor. In addition, the TEC inside the Power Enclosure is connected to the embedded computer over RS232. The general architecture of the system is shown in (Figure 1).

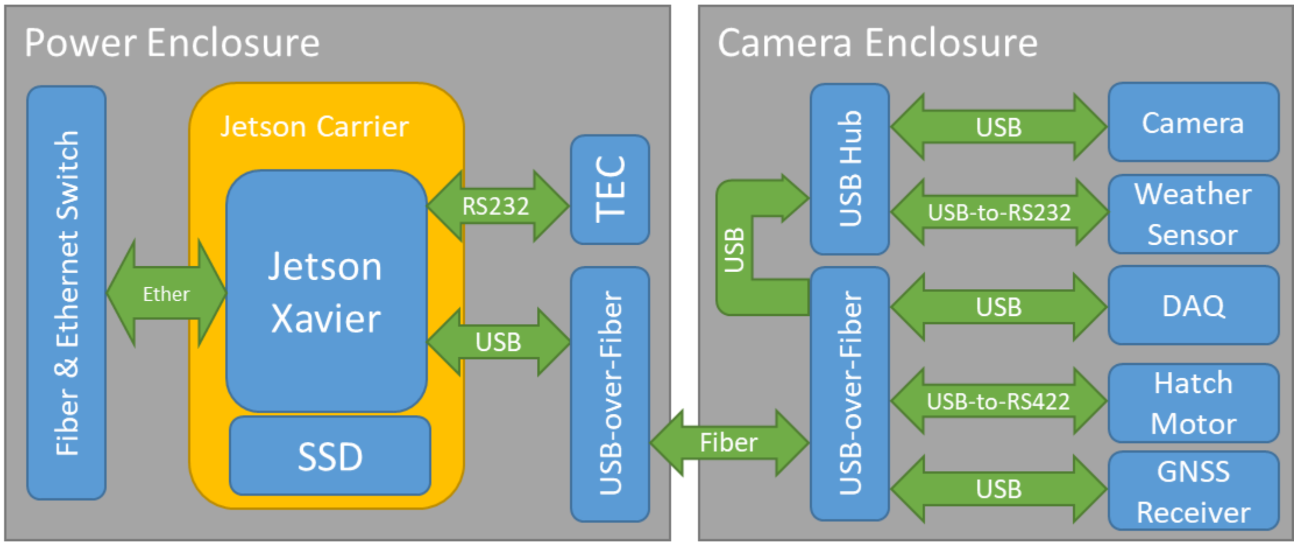


Figure 1. ICI hardware architecture

Application Services

The software is broken up into several services which facilitate different portions of the instrument operations (Table 1). Each of these services starts automatically on boot.

Service description	Service name
Telemetry and hatch control	nwb-ici-telemetry.service
Image capture and processing	nwb-ici-icp.service
Self-characterization	nwb-ici-self-characterization.service
Storage cleanup	nwb-storage-cleanup.service
Weather sensor data collection and storage (optional)	nwb-met-one-weather.service
GNSS data collection and precise positioning solver	nwb-rtklib.service
GNSS precipitable water vapor retrieval	nwb-gnss-pwv-retrieval.service
Precipitable water vapor prediction	nwb-pwv-predictor.service

Table 1. ICI services

The telemetry service continuously collects operational data from the thermoelectric control (TEC) system, the digital acquisition (DAQ) system, the rain sensor, the hatch motor system, and the embedded Jetson platform. It logs these data to a MySQL database. The telemetry service also commands hatch closures dictated by high winds or as commanded by an operator through the web page interface or Python application interface (API). Hatch closures caused by the hatch override button on the Power Enclosure are not commanded by the telemetry service. The data line associated with this button is read directly by the hatch motor. Similarly, the hatch motor reads the rain status data line directly from the rain sensor. In both cases, the telemetry process has visibility into these states but does not command them directly.

The image capture and processing (ICP) service logs camera temperature data to the database, captures images, processes images into cloud products, and saves the data products. The product data for each image are saved as a NetCDF file to an onboard solid-state drive. To ensure that the hatch status is accurately reported in the data products, the ICP service depends on the telemetry service. If the telemetry service fails, the ICP will not run.

The storage cleanup service ensures that the onboard storage never fills by periodically deleting old telemetry and data products. The retention period for each type of data (image products, telemetry, GNSS, etc.) is set within the configuration file.

The self-characterization service runs once a day and attempts to autonomously derive alignment, ground mask, and fixed pattern corrections. It is discussed under the *Instrument Self-Characterization* section of this document.

The weather sensor data collection service periodically collects data from the weather sensor and logs it to the database. When the weather sensor option is not used in the system, this service is disabled.

The GNSS data collection service uses RTKLIB to collect and record raw GNSS observables and location data as well as solve for a precise point positioning (PPP) solution. As part of the precise positioning solution, RTKLIB solves for the zenith tropospheric delay. The zenith tropospheric delay data are transmitted over TCP (port 1024) to the GNSS precipitable water vapor (PWV) retrieval service. The retrieval service uses these data along with current weather conditions to retrieve PWV. The resulting PWV observations have an accuracy of approximately 1.8 mm (RMSE). The PPP solution derived by the RTKLIB service requires real time precise ephemeris and clock data from the Real Time Service (RTS) of the International GNSS Service (IGS) to function. These data are broadcast over the internet. Therefore, the ICI must have an internet connection to allow for PWV retrieval.

Image Capture and Processing Application

The ICP application is the key application which captures images, derives cloud products, and saves the resulting data. Figure 2 shows a simplified flow of the image capture loop.

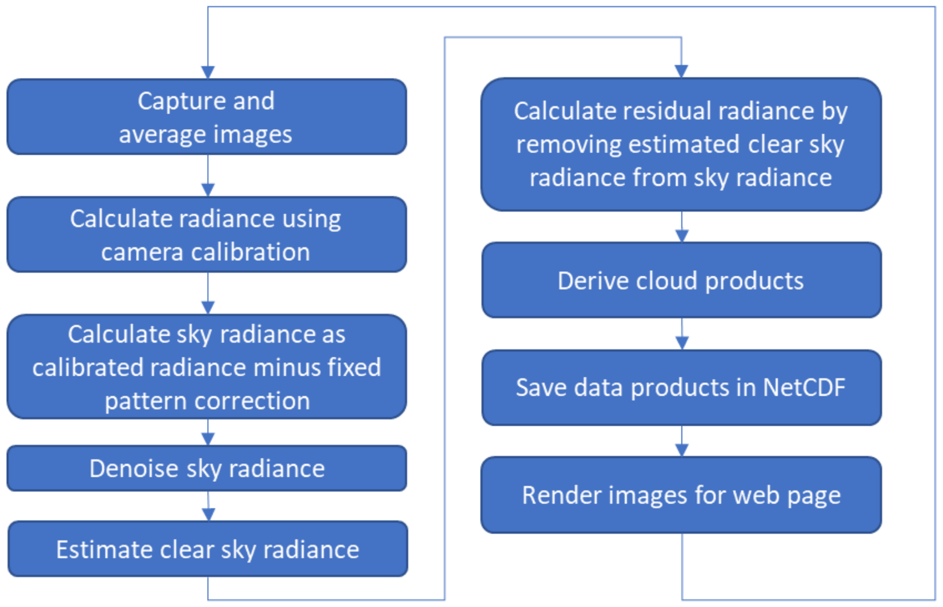

Capture and average images
Calculate radiance using camera calibration
Calculate sky radiance as calibrated radiance minus fixed pattern correction
Denoise sky radiance
Estimate clear sky radiance
Calculate residual radiance by removing estimated clear sky radiance from sky radiance
Derive cloud products
Save data products in NetCDF
Render images for web page

Figure 2. Image Capture and Processing application flow

First, multiple raw images are captured by the camera. Currently, 30 frames are collected over approximately 500 milliseconds. These images are averaged to reduce noise. The camera calibration is then applied to this averaged image to calculate the calibrated radiance image. The sky radiance is calculated as the calibrated radiance minus the current fixed pattern correction which is derived after installation using long term detections of clear sky. In addition, denoising is applied to the sky radiance image to remove any residual column noise from the bolometer.

Then, the clear sky atmospheric radiance is estimated. The clear sky radiance will choose to model the radiance several ways depending on the availability of current PWV and weather data. The decision flow is as follows:

1. If a recent surface weather observation is available and:
    a. If a recent PWV observation is available, the recent PWV and weather observations are used to create the clear sky radiance.
    b. If a recent PWV prediction is available, the recent predicted PWV and weather observations are used to estimate the clear sky radiance.
    c. Otherwise, the weather is used in a surface-only model to estimate the clear sky radiance.
2. If a recent surface weather observation is not available, default values for barometric pressure and humidity are used in the model. Temperature is assigned as follows:
    a. If a temperature climatology is present, time of day and year is used to estimate a temperature.
    b. Otherwise, the default configuration temperature is used.

    PWV is assigned as follows:

    a. If a recent PWV observation is available, it is used in the model.
    b. If a recent PWV prediction is available, it is used in the model.
    c. If a PWV climatology is available, time of year is used to estimate it for the model.

d. Otherwise, the default configuration temperature is used.

Climatology and default values are used as a last resort and are provided only to maintain a minimum level of instrument operation. The resulting residual radiance values and cloud products will commonly be severely degraded under these conditions. For systems which use the weather sensor, the climatology and default values will never be used if the weather sensor continues to report. Only a failure of the weather sensor would cause the system to utilize these fallbacks.

After the clear sky radiance has been estimated, the residual radiance is calculated as the sky radiance minus the modeled radiance.

Cloud probability is derived from residual radiance. The residual radiance is compared to a statistical characterization of the noise and calibration accuracy of the camera. These are used to determine the probability that each pixel of the image contains a cloud. A threshold is applied to this probability to define the cloud mask.

The raw image, cloud products, and select telemetry are then saved to a NetCDF file. After these images are rendered for web page display, the loop delays as necessary to achieve the intended capture interval. Then, the entire image capture and processing loop starts over.

Software Setup

The following software setup requirements must be addressed before operations can proceed.

Meteorological Data

The ICI software requires real-time surface meteorology observations to characterize the sky radiance emitted below clouds. In addition, wind speed observations are used to determine whether blowing debris may be a threat to the system. Under high wind conditions, the system protects itself by closing its hatch. With the standard system, local surface meteorology is provided by an integrated weather sensor. This provides an all-in-one solution for the requisite meteorology. Alternatively, surface meteorology data (temperature, humidity, barometric pressure, and wind speed) can be posted to the system using the application interface.

In addition, the clear sky radiance estimate can be significantly improved with knowledge of the precipitable water vapor (PWV) in the atmosphere. The standard system provides PWV data that is derived from data collected by the integrated Global Navigation Satellite System (GNSS) antenna and receiver. Alternatively, the operator can post external PWV observations to the instrument using the application interface.

See the *Remote Application Interface* section for more information on posting externally observed weather and PWV data to the ICI.

Location Setup

The ICI software uses time and its geodetic location for sun position calculation. With the standard system, a GNSS antenna and receiver system are included to allow the system to autonomously determine its location. To accommodate site security requirements, the operator can elect to remove the GNSS system from the instrument. In this case, the system location must be set manually in the configuration file prior to operation. See the *Operational Parameters* section for configurable location parameters.

Climatology Fallback Setup (optional)

The Image Capture and Processing (ICP) application can utilize climatology for fallbacks in event that weather or PWV data are unavailable. See *Image Capture and Processing Application* for more information.

Two climatology files may be uploaded to the instrument. The first file contains the PWV climatology while the second contains the surface temperature climatology. These files must be externally generated by the user for the installation site and then uploaded to the ICI. The filesystem locations for each of these files are configurable. See the *Operational Parameters* section for the default locations of these files.

Examples of the two climatology types are shown in Figure 3 and Figure 4. These files contain comma-separated values and have the following characteristics:

1. Header rows at the top of the file begin with a # are ignored.
2. After the header, a single label row is expected but ignored.
3. The first data column contains the month numbers 1 through 12.
4. For the PWV file, the second column must contain 12 values in 12 rows corresponding to 12 months starting with the January value after the header row. PWV values must be in millimeters.
5. For the temperature file, columns 2 – 25 contain temperature data for hours 0 – 23 respectively. The 12 rows of data correspond to January – December. Temperature values in the climatology files are in °C.

```
# Monthly average Precipitable Water Vapor (PWV) climatology
# PWV values are given in millimeters (mm)
# Rows are average PWV for each month of the year
MM, PWV (mm)
1,7.54
2,7.58
3,8.27
4,8.39
5,9.75
6,16.94
7,28.21
8,28.27
9,23.68
10,14.05
11,10.47
12,8.19
```

Figure 3. Example PWV climatology file

```
# Hourly mean surface temperature (Celcius) climatology by month
# Rows are each month of the year
# Columns represent the hour of the day in UTC
# Data are mean hourly temperature for each of the 12 months
        00,     01,     02,     03,     04,     05,     06,     07,     08,     09,     10,     11,     12,     13,     14,     15,     16,     17,     18,     19,     20,     21,     22,     23,
01,    8.9,    7.0,    5.8,    4.9,    4.2,    3.6,    3.2,    2.5,    2.2,    1.9,    1.4,    1.0,    0.7,    0.4,    1.9,    5.3,    7.4,    9.1,   10.6,   11.9,   12.6,   13.1,   13.0,   11.7,
02,   13.8,   11.4,    9.8,    8.9,    7.9,    7.2,    6.4,    5.8,    5.2,    4.7,    4.0,    3.7,    3.4,    3.0,    5.9,    9.0,   11.1,   12.9,   14.5,   15.7,   16.6,   17.1,   17.0,   16.0,
03,   18.8,   16.1,   14.4,   13.2,   12.2,   11.4,   10.5,    9.9,    9.1,    8.4,    7.8,    7.3,    6.6,    7.4,   11.2,   13.8,   16.0,   17.6,   19.0,   19.9,   20.6,   20.9,   20.9,   20.2,
04,   23.1,   20.5,   18.5,   17.0,   15.8,   14.7,   13.7,   12.8,   12.1,   11.2,   10.6,    9.8,    9.4,   13.0,   15.9,   18.1,   19.9,   21.5,   22.8,   23.8,   24.6,   25.0,   24.9,   24.3,
05,   27.3,   25.2,   22.6,   20.8,   19.5,   18.5,   17.5,   16.7,   15.7,   14.8,   14.0,   13.3,   14.3,   18.1,   20.6,   22.6,   24.2,   25.6,   27.0,   28.0,   28.7,   29.0,   29.0,   28.3,
06,   33.2,   31.4,   28.6,   26.9,   25.9,   24.8,   23.9,   23.0,   22.1,   21.4,   20.6,   19.9,   21.1,   24.1,   26.4,   28.4,   30.2,   31.6,   33.0,   34.0,   34.7,   35.1,   35.0,   34.3,
07,   31.9,   30.2,   28.2,   27.1,   26.1,   25.2,   24.3,   23.6,   23.1,   22.4,   21.7,   21.3,   21.7,   24.1,   25.9,   27.6,   29.2,   30.8,   32.2,   33.3,   33.9,   34.2,   33.9,   33.1,
08,   30.8,   28.6,   27.1,   26.0,   25.1,   24.2,   23.4,   22.8,   22.1,   21.5,   20.9,   20.3,   20.1,   22.5,   24.5,   26.3,   28.1,   29.6,   30.8,   31.9,   32.6,   32.9,   32.7,   32.1,
09,   27.3,   25.0,   23.6,   22.5,   21.6,   20.9,   20.2,   19.7,   19.0,   18.3,   17.8,   17.4,   17.0,   19.0,   21.8,   23.6,   25.2,   26.5,   27.7,   28.6,   29.4,   29.6,   29.4,   28.9,
10,   21.0,   19.0,   17.8,   16.7,   15.6,   14.9,   14.3,   13.6,   13.1,   12.4,   11.9,   11.4,   10.8,   12.0,   16.1,   18.3,   20.1,   21.9,   23.3,   24.2,   24.7,   25.0,   24.7,   24.0,
11,   13.5,   11.9,   10.6,    9.8,    9.0,    8.5,    7.9,    7.4,    6.9,    6.3,    6.0,    5.6,    5.3,    5.0,    8.8,   11.5,   13.5,   15.2,   16.6,   17.6,   18.2,   18.3,   18.0,   16.5,
12,    8.8,    7.3,    6.3,    5.4,    4.7,    4.3,    3.8,    3.3,    2.9,    2.6,    2.3,    1.9,    1.6,    1.4,    3.1,    6.0,    7.8,    9.7,   11.2,   12.3,   13.0,   13.2,   13.0,   11.5,
```

Figure 4. Example of surface temperature climatology file

Starting the System

For installation instructions and network requirements, refer to the accompanying *All-Sky Infrared Cloud Imager Installation Manual*.

The ICI system starts automatically upon power up. To start the system, perform the following steps:

1. Connect the AC power cable attached to the Power Enclosure to a power outlet.
2. Open the Power Enclosure and ensure that the orange breaker is set to “On”.
3. Confirm that the “DC OK” LED on the TDK-Lambda (DRF-480-24-1) power supply is lit.
4. If deploying in cold weather, the system will wait for the thermoelectric control (TEC) system to warm the Camera Enclosure to ~5 °C before powering up the system. This may take up to 30 minutes.
5. Confirm that the “DC OK” LED on the Phoenix Contact (2902999) power supply is lit. Then, allow 45 seconds for the embedded system to boot. Also, the green LEDs on the power relays should be lit (Figure 5).
6. Under dry conditions, the hatch will open approximately 2 minutes after boot.
7. To verify operation, access the system by connecting to it using one of the software interfaces described in this document.

In the event of a power failure, the ICI will automatically restart when power is restored.

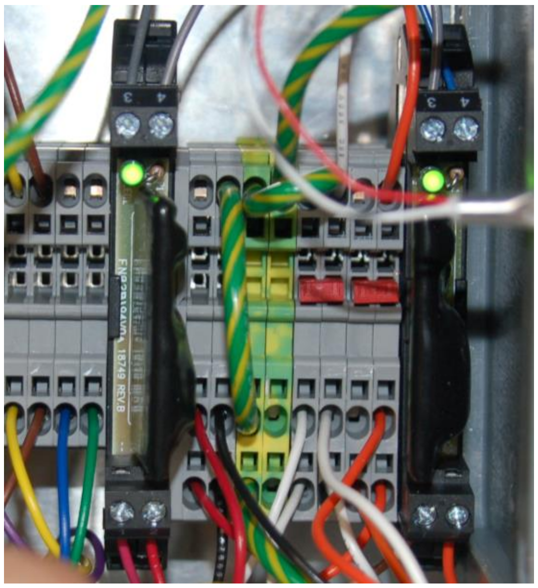


Figure 5. Green LEDs on power relays in Power Enclosure.

Data Collection

Once the system is running, it will continuously collect imagery and save data products. By default, images are collected every minute. Image capture has been tested as fast as a 15-second cadence.

The main configuration file controls the frequency of data collection and the location of local storage. Data are stored to an onboard solid-state disk, which can support operation for up to 100 days at a 15-second cadence, or 400 days at a 1-minute cadence. All data products and database data are automatically deleted from local storage after a retention period. By default, this is set to 150 days and is configurable.

Data products can be downloaded using the application interface.

User Interfaces

Three interfaces are provided for controlling the ICI: the command line interface (CLI), the remote application interface (API), and the web interface. As required, the web interface can be disabled for security considerations. It is not password protected, and any network user can use it to command the instrument.

Image display tabs

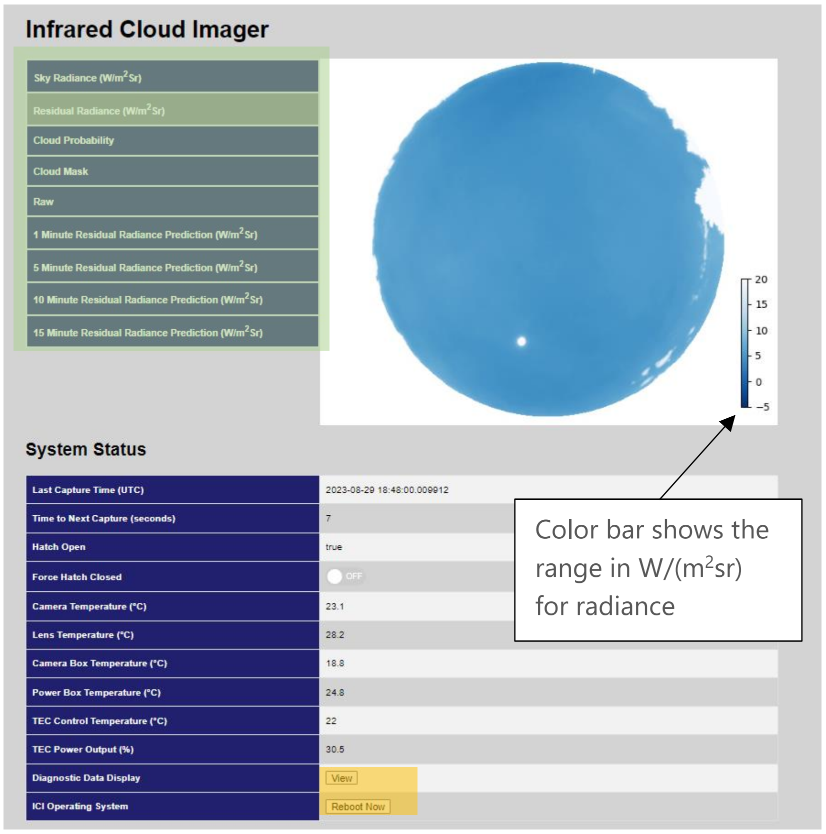
Select a display tab to choose between various products for the most recent image including raw image data, sky and residual radiances, and cloud products.

Controls

Controls for forcing the hatch closed, rebooting the embedded computer, and for opening the Grafana diagnostic debug display.

Telemetry data

Primary status indicators for the camera, hatch, and TEC system are listed first. The most recent weather and atmospheric reports are listed next. Diagnostic data values are listed last to facilitate troubleshooting.


Infrared Cloud Imager
Sky Radiance (W/m²Sr)
Residual Radiance (W/m²Sr)
Cloud Probability
Cloud Mask
Raw
1 Minute Residual Radiance Prediction (W/m²Sr)
5 Minute Residual Radiance Prediction (W/m²Sr)
10 Minute Residual Radiance Prediction (W/m²Sr)
15 Minute Residual Radiance Prediction (W/m²Sr)
20
15
10
5
0
−5
Color bar shows the range in W/(m²sr) for radiance

System Status

Last Capture Time (UTC)	2023-08-29 18:48:00.009912
Time to Next Capture (seconds)	7
Hatch Open	true
Force Hatch Closed	OFF
Camera Temperature (°C)	23.1
Lens Temperature (°C)	28.2
Camera Box Temperature (°C)	18.8
Power Box Temperature (°C)	24.8
TEC Control Temperature (°C)	22
TEC Power Output (%)	30.5
Diagnostic Data Display	View
ICI Operating System	Reboot Now

Figure 6. Screenshot of the ICI web interface

Web Interface

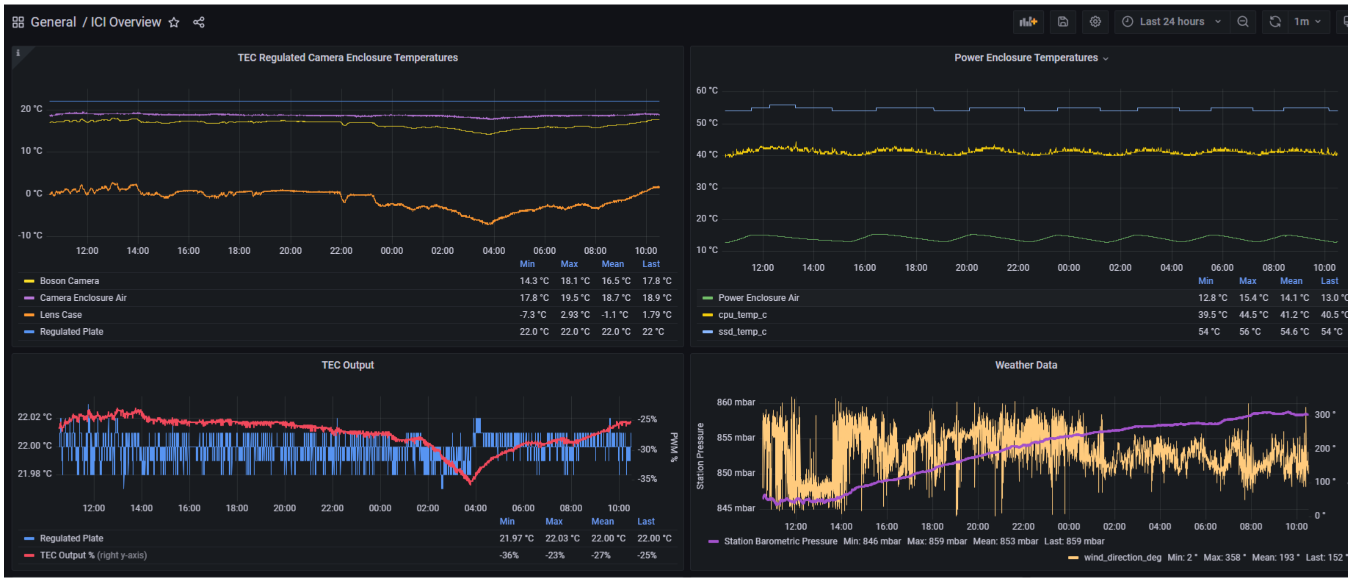
The web interface provides a convenient display of the current system state and products. To access the web interface from a computer connected to the same local area network as the ICI, type *http://<host name or IP address>* (e.g., http://ici-1.local) into a browser window. The website will become available once the system is fully operational. This is about 90 seconds after power is applied. Figure 6 shows a screenshot of the web page and identifies its key components.

The web page has three control buttons for the ICI system.

Restart ICI: Reboots the ICI embedded system. This can be used to reinitialize the system after a calibration or configuration update.

Force Hatch Closed: Select "On" to force the hatch into a closed state. The hatch will remain closed until this flag is cleared. The force hatch closed flag can be set or cleared from the web interface, CLI, or the API. This state will persist through reboots, so an operator must explicitly clear it to allow the system to return to normal operations.

Debug: Opens the Grafana window (see Figure 7) which charts historical telemetry data. This interface is solely provided to allow NWB Sensors to better support the operator with remote troubleshooting. Specific fields are not documented here.



	Min	Max	Mean	Last
Boson Camera	14.3 °C	18.1 °C	16.5 °C	17.8 °C
Camera Enclosure Air	17.8 °C	19.5 °C	18.7 °C	18.9 °C
Lens Case	-7.3 °C	2.93 °C	-1.1 °C	1.79 °C
Regulated Plate	22.0 °C	22.0 °C	22.0 °C	22 °C

	Min	Max	Mean	Last
Power Enclosure Air	12.8 °C	15.4 °C	14.1 °C	13.0 °
cpu_temp_c	39.5 °C	44.5 °C	41.2 °C	40.5 °
ssd_temp_c	54 °C	56 °C	54.6 °C	54 °C

	Min	Max	Mean	Last
Regulated Plate	21.97 °C	22.03 °C	22.00 °C	22.00 °C
TEC Output % (right y-axis)	-36%	-23%	-27%	-25%

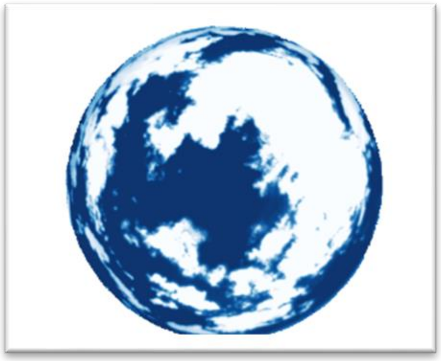
Station Barometric Pressure Min: 846 mbar Max: 859 mbar Mean: 853 mbar Last: 859 mbar
wind_direction_deg Min: 2 ° Max: 358 ° Mean: 193 ° Last: 152 °

Figure 7. Example ICI debug screen.

The latest image and cloud products are displayed at the top of the web page. A series of tabs on the top of the image allows the operator to choose between various image and cloud products. The following data products are available for viewing:

Sky Radiance

Selecting the Sky Radiance tab displays the calibrated sky radiance for display. These values are in units of W/(m^2sr). Data for clear skies to moderately thick clouds are expected to be in the 0 to 25 W/(m^2sr) range. Data for this image are saved in the NetCDF file.



Residual Radiance

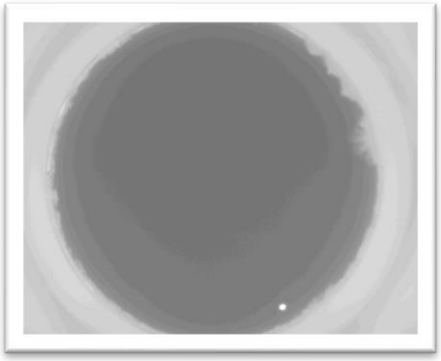
Selecting the Residual Radiance tab displays the residual radiance. These values are in units of W/(m^2sr). These data represent the emitted radiance from clouds after clear sky emission has been removed. The range for these data is set to 0 to 20 W/(m^2sr) by default. Data for this image are saved in the NetCDF file.

Cloud Probability

Selecting the Cloud Probability tab displays the fractional probability (from 0 to 1) that a pixel is a cloudy pixel. These images are calculated from the residual radiance. Data for this image are saved in the NetCDF file.

Cloud Mask

Selecting the Cloud Mask tab displays the cloud mask product (0 = clear sky, 1 = cloud). This mask is calculated from the cloud probability data using the threshold defined in the configuration file. Data for this image are saved in the NetCDF file.



Raw Data

Selecting the Raw tab will display the raw unprocessed image. The image is normalized for display purposes to give a better contrast. This normalization allows clear and cloudy sky to be viewed with a better dynamic range but causes the sun and other hot objects to be saturated.

Command Line Interface

The command line interface (CLI) allows an operator to perform common operations through an SSH console. After logging onto the ICI over SSH, these commands can be executed from any system folder.

Remote Application Interface

The application interface (API) is provided by the IciApi python class (in api.py). It allows an operator's application software to remotely control the ICI and access its data products. The command portion of this class is effectively a wrapper around the CLI SSH commands. The class also provides for downloading the NetCDF data products. The API is based on Python 3 and has been tested under Python 3.8 and 3.9. The following dependencies are required for its use:

Python 3 API dependencies
paramiko
tqdm

Table 2. Remote API Python 3 dependencies

The API software demos are included with the ICI driver code at */usr/local/lib/python3.6/dist-packages/ici_driver/client*. To use them, copy (scp or rsync) the entire client folder to your local host. Each file provides instructions on how to execute. The SSH user and password is provided by NWB Sensors at time of delivery.

CLI and API Commands

For most CLI commands, there is an equivalent method in the API to perform the same task. If the output of a CLI command is JSON, the corresponding API command will output a Python dictionary. The *api_demo.py* script provides example code to remotely command the instrument. This script is a recommended starting point for developers. In the following section, it is assumed that the IciApi class has been instantiated to a variable called *ici* in a similar fashion to the following:

```
# Establish the connection to the ICI
ici = IciApi(host=host_name, username=ici, password=pwd)
```

The CLI and corresponding API commands are documented below.

Force Hatch Closed Command

```
python3 -m ici_driver.command.set_force_hatch_closed flag
```

Sets the software hatch close override.
If *flag* is 1, the hatch is forced closed indefinitely until cleared. (This persists through system reboots.)
If *flag* is 0, any previous hatch closed override is cleared.

Example CLI usage:

```
python3 -m ici_driver.command.set_force_hatch_closed 1
python3 -m ici_driver.command.set_force_hatch_closed 0
```

Corresponding API usage:

```
ici.force_hatch_closed()
ici.allow_hatch_open()
```

Post Weather Observation Command

```
python3 -m ici_driver.command.post_weather_observation json_string
```

Posts an external surface meteorology observation to the database. The *json_string* must have the following parameters:

JSON key	Description	Example
dt	UTC time of observation with format: YYYY-MM-DD hh:mm:ss	2020-09-30 22:28:45
temp_c	Surface temperature in Celsius (float)	12.51
relative_humidity_pct	Relative humidity in percent (float)	69.0
barometric_pressure_mb	Barometric station pressure (not sea level adjusted) in millibar (float)	832.54
wind_speed_mps	Wind speed in meters per second (float)	3.09

Table 3. Surface meteorology post JSON parameters

Example CLI usage:

```
python3 -m ici_driver.command.post_weather_observation '{"dt": "2020-09-30
      22:28:45", "temp_c": 12.51, "relative_humidity_pct": 69.0,
      "barometric_pressure_mb": 832.5450067594081, "wind_speed_mps": 3.09}'
```

Corresponding API usage:

```
weather = dict()
weather['dt'] = datetime(year=2020, month=9, day=30, hour=22, minute=28, second=45)
weather['temp_c'] = 32.3
weather['relative_humidity_pct'] = 35
weather['barometric_pressure_mb'] = 830
weather['wind_speed_mps'] = 0.53
ici.post_weather_observation(weather)
```

Post Precipitable Water Vapor Observation Command

`python3 -m ici_driver.command.post_pwv_observation` *`json_string`*

Posts an external PWV observation to the database. The *json_string* must have the following parameters:

JSON key	Description	Example
dt	UTC time of observation with format: YYYY-MM-DD hh:mm:ss	2022-08-10 12:00:00
pwv_cm	Columnar integrated precipitable water vapor in centimeters (float)	2.3

Table 4. Precipitable water vapor post JSON parameters

Example CLI usage:

```
python3 -m ici_driver.command.post_pwv_observation '{"dt":"2022-08-10
      12:00:00","pwv_cm":2.3}'
```

Corresponding API usage:

```
atmosphere = dict()
atmosphere['dt'] = datetime(year=2020, month=1, day=1, hour=12, minute=0)
atmosphere['pwv_cm'] = 2.3
ici.post_pwv_observation(atmosphere)
```

Get Telemetry Command

`python3 -m ici_driver.command.get_telemetry`

Returns the most recent telemetry observations for each subsystem. These are encoded as JSON encoded telemetry for the CLI or as a python dictionary for the API. In Table 5, root level keys are listed in bold. Nested field keys are listed below each root key.

Example CLI usage:

```
python3 -m ici_driver.command.get_telemetry
```

Corresponding API usage:

```
ici.get_telemetry()
```

Key	Description
station	
latitude_deg	Station latitude in decimal degrees ranging from -90 to 90 where negative is South. This is sourced from the GNSS receiver. For systems without the GNSS system option, the configuration file sets this value.
longitude_deg	Station longitude in decimal degrees ranging from -180 to 180 degrees where negative is West. For systems without the GNSS system option, the configuration file sets this value.
altitude_m	Station altitude from sea level in meters. For systems without the GNSS system option, the configuration file sets this value.
software	
ici_driver_version	Git hash code of the commit used for the ici_driver repository
ici_web_app_version	Git hash code of the commit used for the ici web app repository
service	
nwb_ici_icp_uptime_sec	Uptime in seconds of the ICP service
nwb_ici_telemetry_uptime_sec	Uptime in seconds of the telemetry service
nwb_met_one_weather_uptime_sec	Uptime in seconds of the Met One weather sensor service
nwb_storage_cleanup_uptime_sec	Uptime in seconds of the storage cleanup service
nwb_gnss_pwv_uptime_sec	Uptime in seconds of the GNSS PWV retrieval service
nwb_pwv_predictor_uptime_sec	Uptime in seconds of the PWV predictor service
ntp_synchronized_status	If "yes", the clock has been synchronized successfully to a NTP server.
hatch	
is_open	True if the hatch is open
is_raining	True if the rain sensor detects rain
is_high_wind	True if a high wind observation has tripped the hatch close according to the rules in the configuration file
is_reed_switch_tripped	True if the hatch magnet is tripping reed switch which indicates that the hatch is closed.
is_override_button_pressed	True if the hatch override button on the Power Enclosure has been pushed.
is_forced_closed	True if the CLI, python API, or web interface has commanded the hatch to be forced shut.
motor_position	Motor encoder step position. Negative steps are toward a hatch open.
camera	
fpa_temp_c	Camera focal plane temperature in Celsius
last_capture_time	Date and time of last capture in UTC with format: YYYY-MM-DD hh:mm:ss.ffffff

seconds_to_next_capture	Number of seconds until the next image is captured
daq	
camera_box_temp_c	Camera box air temperature in Celsius
lens_temp_c	Lens temperature as measured at the focus ring in Celsius
supply12_v	Measured voltage of the 12 V supply
supply24_v	Measured voltage of the 24 V supply
rain_analog_v	Rain sensor analog voltage. When not raining, the voltage should be near 3.0 V
rain_state_v	DAQ observed voltage of the rain sensor status line. Near 0 V means it is raining. This differs from the hatch.is_raining field in that it is read by the DAQ instead of the hatch motor. The states indicated by both should agree.
hatch_state	DAQ observed hatch state (0 is open, 1 is closed) This differs from the hatch.is_open field in that it is read by the DAQ instead of the hatch motor. The states indicated by both should agree.
hatch_reed_switch_v	Voltage of the reed switch. When the hatch is open, the reed switch voltage will be near 2.5 V. When the hatch is closed, it will be near 0 V.
hatch_override	Hatch override button state (0 if button is pressed)
force_close_hatch	1 if web interface, CLI, or API has commanded the hatch to be forced closed. 0 if no software interface has commanded the hatch to be closed.
is_rain_heater_enabled	1 if rain sensor heater is activated. 0 if rain sensor heater has been deactivated due to dangerously low outdoor air temperatures which can burn out the heater.
is_external_fans_enabled	1 if external fans are powered, 0 otherwise. Fans are configured to activate with high TEC utilization.
tec	
power_box_temp_c	Power Enclosure temperature in Celsius
control_temp_c	The temperature in Celsius of the TEC system thermistor used for temperature control. This is located on the camera's cold finger.
power_output_pct	TEC system utilization as a percentage. Negative values indicate that the system is heating.
platform	
uptime_sec	Seconds since the last Linux operating system boot
cpu_load_pct	CPU load as a percentage
ram_utilization_pct	RAM utilization as a percentage
cpu_temp_c	CPU temperature in Celsius
ssd_temp_c	Solid state disk temperature in Celsius
root_free_space_mb	Root filesystem (Jetson flash storage) free space in megabytes
ssd_free_space_mb	Solid state disk free space in megabytes
weather	
dt	Latest surface weather observation time in UTC with format: YYYY-MM-DD hh:mm:ss
temp_c	Temperature in Celsius
relative_humidity_pct	Relative humidity in percent
barometric_pressure_mb	Station barometric pressure in millibar
wind_speed_mps	Wind speed in meters per second
wind_direction_deg	Wind direction in degrees from North
precipitable_water_vapor	

dt	Latest PWV observation time in UTC with format: YYYY-MM-DD hh:mm:ss
pwv_cm	Precipitable water vapor in centimeters
predicted_precipitable_water_vapor	
dt	Latest PWV prediction time in UTC with format: YYYY-MM-DD hh:mm:ss
pwv_cm	Latest PWV predicted in centimeters
zenith_total_delay	Zenith total delay (ZTD) is the estimated total delay of the GNSS signals at zenith as derived by the GNSS processing software. The GNSS-derived precipitable water vapor is derived from this value. This is only typically used for GNSS PWV-retrieval system diagnostics.
dt	Latest zenith total delay observation time in UTC with format: YYYY-MM-DD hh:mm:ss
ztd_cm	Latest zenith total delay in centimeters

Table 5. Telemetry descriptions

Flash Motor Software Command

WARNING: Do not attempt to execute this command unless directed by NWB Sensors.

```
sudo python3 -m ici_driver.command.flash_hatch_motor motor_script_file
```

Flashes a new embedded software program to the hatch motor where *motor_script_file* is the path to the *.mxt* motor file.

Example CLI usage:

```
sudo python3 -m ici_driver.command.flash_hatch_motor /home/ici/ici-motor-
controller/ICI_2022_v2.mxt
```

Dump Meteorology Command

```
sudo python3 -m ici_driver.command.dump_meteorology
```

Dumps all weather and PWV observations and predictions stored in the MySQL *weather* and *precipitable_water_vapor* tables to an SQLite database. These data are commonly extracted with NetCDF files to allow post-processing of the raw image data later. The maximum history stored in these tables is defined by the data retention period in the configuration file (which defaults to 90 days).

The database file is stored to /storage/. The name will have the following format:

meteorology_dump.yyyy-mm-dd_HHMMSS.sqlite

Example CLI usage:

```
sudo python3 -m ici_driver.command.dump_meteorology
```

Dump Diagnostic Data Command

```
sudo python3 -m ici_driver.command.dump_diagnostic_data
```

Dumps all database tables and system logs into a compressed archive (this may take a long time). In the event that the system has problems, NWB Sensors will request that you run this command, downlink the resulting archive, and send it for evaluation.

The resulting file is stored to /storage/. The name will have the following format:

dump.yyyy-mm-dd_HHMMSS.tar.gz

Example CLI usage:

```
sudo python3 -m ici_driver.command.dump_diagnostic_data
```

Downloading Data Products through the API

See the *downloader_demo.py* and end of the *api_demo.py* files in the provided API code for examples of downloading NetCDF output files generated over the last five minutes. To download NetCDF data products through the API, a file list is first queried for the time of interest. The resulting list of absolute file paths is sorted by time. Then, this list is passed to the download method which downloads the images from the ICI to the remote computer. This allows an operator's remote application to autonomously pull data products from the ICI. By default, files that have already been downloaded are ignored. See the associated python class documentation.

Example API usage:

```
# Establish the connection to the ICI
ici = IciApi(host=host_name, username=ici, password=pwd)
# Find paths for all files in the last 5 minutes
start = datetime.utcnow() - timedelta(minutes=5)
files = ici.find_output_files(start)
# Download all files to a local temp directory
temp_dir = Path(tempfile.gettempdir()) / 'ici-download-test'
ici.download_files(files, temp_dir)
```

Alternatively, a remote operator could use a `rsync` command to synchronize the output storage folder (see *Filesystem Locations*) to a folder on an external system. For some use cases (especially external applications which don't use the Python API), this may be a more attractive option.

Setting up passwordless SSH and API access

For security reasons, it is preferrable to eliminate passwords when using the Python API, so that passwords don't have to be stored with calling code. The Python API operates over SSH, so setting up passwordless SSH will also allow for passwordless API access to the ICI. Setting up passwordless SSH is similar across modern Linux distributions. The following instructions have been tested on an Ubuntu host, but similar instructions will work for other Linux distributions. In these instructions, YOUR_KEY_NAME is something you pick. It is the file name of the SSH key pair. It should be something meaningful like "ici-access-key", and ICI_HOST_NAME is the computer host name of the ICI which is shown at the prompt when you log into it (e.g. ici-1).

```
# On your host, generate the private and public key pair.
ssh-keygen -t ed25519 -b 4096 -f <YOUR_KEY_NAME>
# Don't use a pass phrase when prompted.

# Add the key for your host.
ssh-add ~/<YOUR_KEY_NAME>

# Install the public key on the ICI.
ssh-copy-id -i ~/<YOUR_KEY_NAME> ici@<ICI_HOST_NAME>
# Provide the ICI's access password given to you by NWB at the prompt.
# Depending on how your host is setup, you may need to start the SSH agent to use
the new keys as follows:
eval $(ssh-agent)

# You should now be able to connect to the ICI without a password
ssh ici@<ICI_HOST_NAME>

# Now the Python API should work without a password.
# In older versions of the API, the password was a required parameter.
# Just set it to an empty string as follows:
ici = IciApi(host="ici-1", username="ici”, password="")
```

Data Products

During operation, the system saves raw data and derived products to NetCDF files. Naming is: [name]yyyymmddHHMMSS.nc where [name] is set by the *net_cdf_prefix* configuration parameter. (See *the Downloading Data Products through the API* section for more information.) When the hatch is closed, the data product files continue to be generated, but the derived product fields (sky radiance, residual radiance, cloud mask, etc.) are filled with dummy values as described below. The following table provides the variables saved in each NetCDF file.

Descriptive Name	Variable Name	Type	Dimension	Units
Raw Sky Image	sky_raw	uint16	640 x 512	DN
Image representing the raw data collected from the LWIR camera. Values are digital numbers with higher counts representing brighter objects.				
Calibrated Sky Radiance	calibrated_radiance_wpm2sr	float32	640 x 512	W/(m^{2}sr)
Optional output parameter which can be activated by the *save_calibrated_radiance* configuration parameter. Radiance derived from the camera calibration with no fixed pattern correction or denoising. Values are in W/(m^{2}sr) as modeled over the camera's spectral response.				
Sky Radiance	sky_radiance_wpm2sr	float32	640 x 512	W/(m^{2}sr)
Sky radiance image which is calculated as the calibrated radiance minus the fixed pattern correction, and with denoising applied. Values are in W/(m^{2}sr) as integrated over the camera's spectral response. Regions outside the FOV are filled with NaNs. Populated with -9999 values when the hatch is closed.				
Modeled Sky Radiance	modeled_clear_sky_radiance_wpm2sr	float32	640 x 512	W/(m^{2}sr)
Optional output parameter which can be activated by the save*_modeled_clear_sky_radiance* configuration parameter. Modeled clear sky emission for each pixel. Values are in W/(m^{2}sr) as modeled over the camera's spectral response.				
Residual Sky Radiance	residual_sky_radiance_wpm2sr	float32	640 x 512	W/(m^{2}sr)
Remaining sky radiance after the modeled clear sky emission has been subtracted from the sky radiance image. Values are in W/(m^{2}sr) as integrated over the camera's spectral response. Regions outside the FOV are filled with NaNs. Populated with -9999 values when the hatch is closed.				
Cloud Probability	cloud_probability_fraction	float32	640 x 512	Fraction
Image representing the probability that a particular pixel is covered by cloud. The range of these data are 0 (not a cloud) to 1 (likely a cloud) as statistically estimated by the processing. Regions outside the FOV are filled with NaNs. Populated with 1’s when the hatch is closed.				
Cloud Mask	cloud_mask	Boolean	640 x 512	None
Binary mask image that represents cloud detection. (0 = not a cloud, 1 = cloud). Populated with 1’s when the hatch is closed. This mask is generated using the cloud_confidence_threshold set in the ici_config.yml file.				
Camera Serial Number	sn	Int64	1	None
Serial number of the FLIR Boson camera				

Camera FPA Temperature	fpa_temp_c	float32	1	Celsius
The temperature in Celsius of the camera's focal plane array when the image was collected.				
Lens Temperature	lens_temp_c	float32	1	Celsius
The temperature in Celsius of the lens when the image was collected.				
Image Collection Time	img_time	int64	1	seconds
Time of image capture in seconds since the Unix epoch (1970-01-01 00:00:00 UTC).				
Sun Mask	sun_mask	Boolean	640 x 512	None
Sun mask image with pixels set to 1 where the sun was detected and excluded from the data processing. Populated with 0's when the hatch is closed.				
Camera Calibration Hash (MD5)	calibration_file_md5	str	1	None
The MD5 hash of the camera calibration file used that was used in the data processing as a hexadecimal ASCII string.				
Columns in image products	cols	int16	1	None
Columns in the raw image and resulting image data products				
Rows in the image products	rows	int16	1	None
Rows in the raw image and resulting image data products				
Is Hatch Closed	is_hatch_closed	Boolean	1	None
Flag representing the hatch state (1 = hatch closed, 0 = hatch open)				
Is Raining	is_raining	Boolean	1	None
Flag representing whether rain is causing a hatch closure (1 = rain, 0 = no rain).				
Is High Wind Condition	is_high_wind	Boolean	1	None
Flag representing whether high wind is causing hatch closure (1 = high wind, 0 = no high wind)				
Is Hatch Override Button Pressed	is_override_button_pressed	Boolean	1	None
Flag representing whether hatch override button on power enclosure is being pressed and causing hatch closure (1 = button pressed, 0 = button not pressed)				
Is Hatch Forced Closed	is_forced_closed	Boolean	1	None
Flag representing whether software command is causing hatch closure (1 = software commands hatch closed, 0 = no software command has commanded hatch closure)				

Table 6. NetCDF products

Operations

Routine Operations Checks

NWB Sensors recommends that the software operator check the system a minimum of once per week. During the checkout, the following procedures should be performed.

- Check for acceptable operational conditions
- Check the imagery for water spots or debris

Routine hardware maintenance tasks are documented in the *All-Sky Infrared Cloud Imager Hardware Maintenance Manual*. Refer to this manual for instructions on:

- Checking and changing the desiccant
- Check the rain sensor and hatch operation
- Cleaning the lens

Acceptable Operating Conditions

During each operations check, confirm that the ICI is operating correctly by assessing the condition of each operating parameter shown in Table 7 through one of the user interfaces.

Parameter	Acceptable condition
Hatch	Hatch should be open unless one of the following is true: • Operator has forced hatch closed. • The Power Enclosure hatch override button has been pressed. • High winds have recently (by default 15 minutes) been detected. • It is raining.
Last Capture Time	When the hatch is open, ensure the last capture occurred recently (within the image capture cadence set in the configuration). Times are given in UTC.
Latest Weather Observation	Ensure the latest weather observation is recent. If the standard weather sensor is being used and weather data are stale (older than 2 minutes), contact NWB Sensors. There may be an issue with the weather sensor. If external weather data are being sent to the instrument, a stale observation indicates a problem with the operator's external application or network connection.
Latest PWV Observation	If sending external precipitable water observations, ensure the latest weather observation is recent. If it is stale, there may be a problem with the operator's external application.
Acceptable Temperatures	Camera FPA: 15 - 40 °C Camera Enclosure: 10 - 30 °C Power Enclosure: 0 - 35 °C CPU: less than 60 °C SSD: less than 70 °C If any temperatures are outside of this range during normal operations, contact NWB Sensors to ensure system components will not be damaged.
TEC Power Output	If this is ±100%, the TEC system is fully utilized. This is expected immediately after installation but should not be observed if the system has been operating for a long period of time. If this occurs over the course of normal operations, you should contact NWB Sensors to assess the situation.
Linux Uptime	During normal operation, the Linux embedded system does not restart. If the uptime indicates that restarts are occurring often, there may be a hardware or software problem. Check the system logs. The telemetry, ICP, and weather services will reboot the system if it fails too many times. This may indicate a problem with the camera hardware. See the *Service Failures and Automatic Recovery* section.
Service Uptime	The software consists of several services which run the instrument: image capture (nwb-ici-icp), telemetry (nwb-ici-telemetry), storage cleanup (nwb-storage-cleanup) and optionally the met-one weather station (nwb-met-one-data-collection). During normal operation, these services should not restart. If any of these services appear to be restarting regularly, it indicates a hardware or software problem with the instrument.

NTP Synchronization Status	Network Time Protocol (NTP) is required for the system to properly timestamp images and perform sun alignment. If the system loses synchronization, the NTP server or the network connection may have issues.
Root Partition Free Space	The root partition has limited free space. It should never drop below 100 MB of free space. This likely indicates that a user is storing too much data in their home directory. Clean up as necessary.
SSD Free Space	The solid-state disk (SSD) is 1 TB. Generally, it should have more than 50 GB of free space. If free storage drops below this level, there may be a problem with the storage cleanup service which deletes stale data automatically. Alternatively, the operator may have stored too much data on this drive.

Table 7. Acceptable operating conditions

Checking for Water Spots or Debris

Water spots or debris will show up as persistent artifacts in clear sky radiance and residual radiance images. These can be viewed using the Web interface or by parsing the NetCDF data products. Figure 8 shows an example of a normal image and examples of images with water spots. If water spots or debris are observed, the imagery should be re-evaluated after an hour of rain-free operation. If the water spots persist, the window should be cleaned according to the instructions in the *All-Sky Hardware Maintenance Manual*.

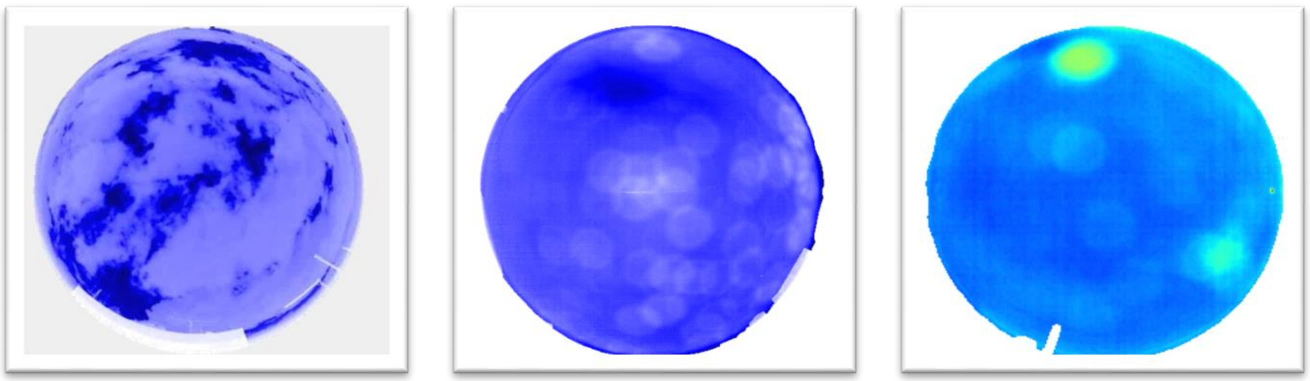


Figure 8. Example images showing water spots in the sky radiance image.

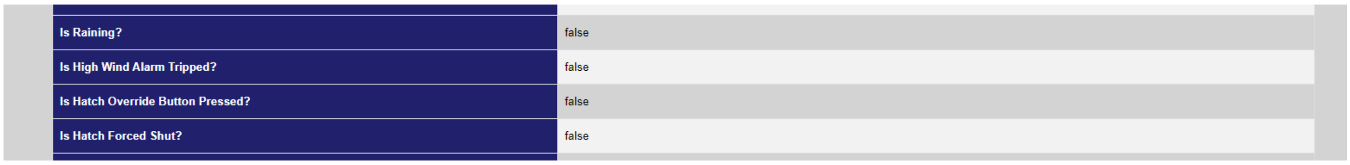
Contending with Extremely High Winds

If winds are expected to reach 70 mph, the hatch should be closed and locked until wind speeds relent. See the *All-Sky Infrared Cloud Imager Hardware Maintenance Manual* for the hatch lock-down procedure.

Diagnosing Hatch Closures

Several different conditions can cause the system to close its hatch: operator intervention through a software interface (Web, CLI, or API), activating the hatch override button on the Power Enclosure, rain, and high winds. Diagnosing a hatch closure condition can be performed through (1) the web interface, (2) the CLI, or (3) the API.

With the web interface, the Force Hatch Close selector (see Figure 6) should first be checked to confirm whether any software interface (Web, CLI, or API) has been used to force the hatch closed. If this is not the cause of the closure, scroll down the web page to the Diagnostics section. The other fields of interest will be shown as seen in Figure 9. With the CLI or API interfaces, see the corresponding fields in the *get_telemetry* command output (see page 18).



Is Raining?	false
Is High Wind Alarm Tripped?	false
Is Hatch Override Button Pressed?	false
Is Hatch Forced Shut?	false

Figure 9. Web page hatch closure diagnostics

Accessing System Logs

As the ICI is running, the full operation log is stored in the system journal. To view the log entries in real-time, connect to the system over SSH. Then, execute the following command:

```
sudo tail -f /var/log/syslog
```

You will see a running log of entries from all processes including those that run the ICI as shown in Figure 10.

```
Nov 11 18:27:16 ici-2 python3[5253]: INFO    [236113.270 ici_telemetry.py:165] Hatch override button pressed: N
Nov 11 18:27:16 ici-2 python3[5253]: INFO    [236113.271 ici_telemetry.py:166] Hatch closed: N
Nov 11 18:27:16 ici-2 python3[5253]: INFO    [236113.271 ici_telemetry.py:167] Hatch reed switch (V): 2.93
Nov 11 18:27:16 ici-2 python3[5253]: INFO    [236113.272 ici_telemetry.py:168] Close hatch (software override): N
Nov 11 18:27:16 ici-2 python3[5280]: INFO    [236110.033 ici_icp.py:172] Weather sensor air temperature: -3.6 C
Nov 11 18:27:16 ici-2 python3[5253]: INFO    [236113.455 ici_telemetry.py:177] Motor position: -13655
Nov 11 18:27:16 ici-2 python3[5253]: INFO    [236113.455 ici_telemetry.py:178] Motor input 1 (reed switch tripped): 0
Nov 11 18:27:16 ici-2 python3[5253]: INFO    [236113.455 ici_telemetry.py:179] Motor input 2 (rain detected): 0
Nov 11 18:27:16 ici-2 python3[5253]: INFO    [236113.455 ici_telemetry.py:180] Motor input 3 (hatch closed): 0
Nov 11 18:27:16 ici-2 python3[5253]: INFO    [236113.455 ici_telemetry.py:181] Motor input 4 (close hatch): 0
Nov 11 18:27:16 ici-2 python3[5253]: INFO    [236113.455 ici_telemetry.py:182] Motor input 5 (hatch override button pressed):  0
Nov 11 18:27:16 ici-2 python3[5280]: INFO    [236110.081 ici_icp.py:196] Using Reitan model for pwv: 0.25 cm
Nov 11 18:27:16 ici-2 python3[5280]: INFO    [236110.082 ici_icp.py:206] Processing image
Nov 11 18:27:16 ici-2 python3[5253]: INFO    [236113.807 ici_telemetry.py:208] Power box temperature (C): 15.6
Nov 11 18:27:16 ici-2 python3[5253]: INFO    [236113.808 ici_telemetry.py:209] TEC control temperature (C): 22.0
```

Figure 10. Example syslog output

Log entries are typically self-documenting. Table 8 highlights log entries of interest:

Entry text example	Explanation
Captured image at 2022-11-11 18:44:02.829695	The camera finished capturing and averaging the raw images before running the cloud processing algorithm. (Multiple captures are averaged to reduce noise.)
Saving /storage/output/data/2021-08-19/ici_20210819001730.nc	The cloud processing algorithm has completed successfully, and the output is saved to a NetCDF file.
Received weather report	Indicates that the application received a weather report from an external application over the API (or CLI).
Received PWV report	Indicates that the application received a precipitable water vapor report from an external application over the API (or CLI).
Read weather sensor	Successfully read the weather sensor.

Table 8. Log entries of interest

Alternatively, specific services can be monitored using the *journalctl* command. See the *Common SSH Commands* section for examples.

Assessing Time Synchronization

Accurate embedded system time is essential to the accuracy of the ICI cloud products. The ICI synchronizes system time with a Network Time Protocol (NTP) server. NTP synchronization status is reported in both the web and API telemetry. From the command line, time synchronization via the NTP service can be checked using the following command:

```
timedatectl status
```

If NTP is not synchronized, execute the following commands to view output from the NTP service. The resulting output can be used to diagnose NTP communication.

```
sudo systemctl status systemd-timesyncd
sudo journalctl -f -u systemd-timesyncd
```

By default, the system uses *pool.ntp.org* servers. See the *NTP Server Configuration* section for information on setting up a local NTP server.

Service Failures and Automatic Recovery

All services described previously will restart after 60 seconds if a failure occurs. The telemetry, ICP, weather sensor, and GNSS PWV retrieval services are considered critical to the operation of the instrument. For these services, the system attempts to recover automatically. If any of these services fail to start itself 5 times within 6 minutes, the embedded computer will reboot its Linux operating system to attempt to autonomously resolve the error condition. If the embedded computer starts rebooting itself repeatedly, there is a persistent issue with the hardware or software that requires outside intervention. If this occurs, contact NWB Sensors for support.

Restarting the software

Occasionally, it may be necessary to restart the ICI software to update configuration parameters, software, or resolve unforeseen errors. Theoretically, each ICI service can be restarted individually as outlined in the *Common SSH Commands* section below, but this is not recommended. Since the ICI is built on multiple services which each read the configuration, it is best to perform a system reboot to eliminate any concern that one or more of the services are using the previous version of the software or configuration. A simple reboot command should be executed from an SSH console prompt as follows:

sudo reboot

The system daemon which automatically starts the services on boot will also gracefully shutdown each of the ICI services during a reboot. It is not necessary or recommended to individually shut down services.

Instrument Self-Characterization

During operation, the instrument autonomously characterizes its deployment environment in two ways. First, it attempts to determine its alignment to the local coordinate frame. Second, it attempts to find a mask of all ground-based objects.

The alignment product consists of azimuth and elevation maps in the local coordinate frame (East-North-Up). Alignment is performed by first detecting the sun position in images over time. Once a set of sufficiently different sun positions have been detected, the detected sun locations are used to determine the rotation of the camera to the local frame. The alignment algorithm determines whether sufficient sun detections have been found to determine the alignment. Typically, a full day of data—with the sun visible throughout the day—is necessary for the first alignment to be derived. Then, the alignment will continue to be updated and improved as subsequent detections are made. As part of the alignment derivation process, any potential shift of the image on the focal plane is measured. Small shifts—typically less than 2 pixels—in image position may result from shipping. The alignment algorithm attempts to detects these and compensate for them.

The ground mask provides an automatically generated mask of detected ground objects. This mask can be utilized by the end user in external processing code. The derived ground mask is not critical to the operation of the instrument, but objects identified in the ground mask are never labeled as clouds in the cloud probability or cloud mask products. The ground mask is derived at the same time as the fixed pattern correction (discussed below).

In addition to the alignment and ground mask products, the instrument also attempts to derive a fixed pattern correction. This correction is necessary to compensate for low-level spatial non-uniformities and bias errors resulting from the camera calibration. Typically, the end user does not make use of the fixed pattern correction, because it is derived during instrument assembly and applied during instrument testing. Under special circumstances, the user may make use of this data product at the direction of NWB Sensors.

The fixed pattern correction algorithm (as part of the self-characterization script) first finds times where the observed radiances across the entire sky agree with those predicted by the clear sky radiance model—within a reasonable level of error. From these times, a median error is calculated on a per-pixel basis across all the detected clear sky images. The period over which clear skies are detected is currently set to three weeks but is configurable. The median, as opposed to a mean, was selected for two reasons. First, it eliminates clouds which may exist in some images. There may be images where small or thin clouds cause only a small deviation from the clear sky model. In these situations, these images may be selected as having a clear sky. The median tends to eliminate these occurrences as outliers. Second, the median serves to average the clear sky model radiance errors which wander over time. The model is limited in its ability to predict the sky radiance. Its accuracy (1-σ) is approximately ~1.2 W/(m^2Sr). This limitation results primarily from variation in the vertical water vapor profile which is not captured with the precipitable water vapor metric. By taking the median of the error over many clear sky detections across several weeks of data, these errors associated with the clear sky model tend to be averaged.

After the fixed pattern correction is successfully run, the ground mask is determined by selecting regions of the image where the observed radiances consistently depart from the radiances predicted by the clear sky model during the selected clear sky periods. Large departures in the radiance only occur when objects that are not clear sky, such as trees or buildings, are in the image. Therefore, the ground mask derivation depends on the fixed pattern correction being successfully derived—whether the associated fixed pattern correction is used or not.

Upon successful execution, the self-characterization algorithm outputs the data products (described in the next sections) to the self-characterization output folder, */storage/self_characterization*, onboard the instrument.

Alignment and Ground Mask Data Products

The self-characterization creates the sky map file which contains both the local coordinate frame maps, and the ground mask. Download the sky map file, *sky_azimuth_elevation_map.nc*, from */storage/self_characterization/*. This NetCDF file contains three variables which can be used in external software:

azimuth_deg: Contains the local azimuth value for each pixel in the image in degrees. North is zero degrees which increases positively to the East. Pixels outside the field of view are mapped as NaNs. This array is the same size as a raw image.

zenith_deg: Contains the local zenith value for each pixel in the image in degrees. Pixels outside the field of view are mapped as NaNs. This array is the same size as a raw image.

ground_mask: Contains a binary mask where ground pixels are 1 and sky pixels are zero. This array is the same size as a raw image.

Fixed Pattern Correction Data Product

The fixed pattern correction can be sanity checked by viewing the *new_calibration.png* file in the self-characterization output folder. An accurate fixed pattern correction will show no indication of cloud contamination. Errors will be smooth. A noticeable ring structure—associated with the calibration process—is commonly exhibited as shown in Figure 11.

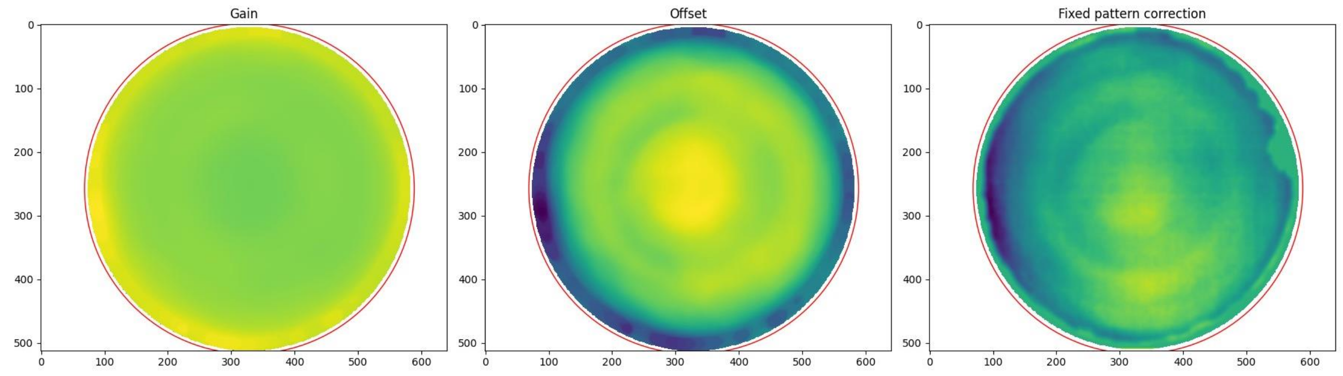


Figure 11. Example of a calibration with a typical fixed pattern correction which only contains calibration artifacts.

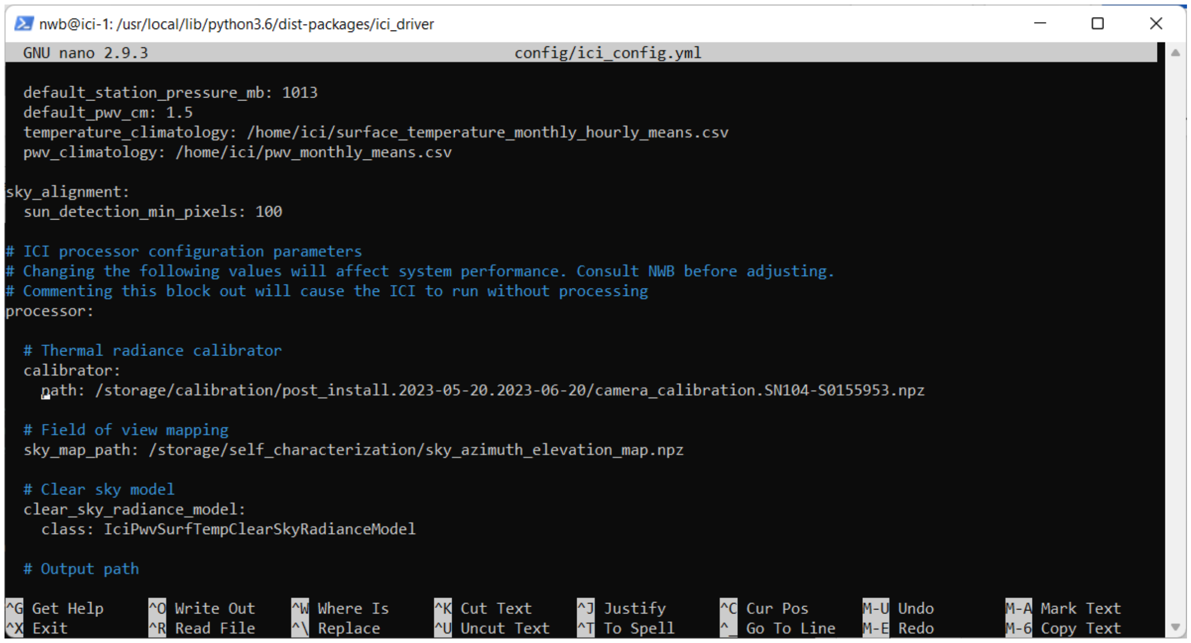
Once a fixed pattern correction is derived, the algorithm will output a new calibration file with the included fixed pattern correction. This new calibration file will have the same name as the original calibration file (stored in */storage/calibration*) but with an *.fpc.npz* suffix.

Making Use of the Fixed Pattern Correction Data Product

NWB Sensors recommends not utilizing the fixed pattern correction product without consulting with them first. To use this file, first copy it from the */storage/self_characterization/* folder to the

/storage/calibration/ folder and update the calibration path which is set under the processor.calibrator.path key in the ICI configuration file. To update the configuration file, a console-based editor (such as nano or vi) must be used with super user privileges. For example, the following command could be executed to open the file in nano. Be sure to save changes when done.

```
sudo nano /usr/local/lib/python3.6/dist-packages/ici_driver/config/ici_config.yml
```



```
nwb@ici-1: /usr/local/lib/python3.6/dist-packages/ici_driver
 GNU nano 2.9.3                          config/ici_config.yml

  default_station_pressure_mb: 1013
  default_pwv_cm: 1.5
  temperature_climatology: /home/ici/surface_temperature_monthly_hourly_means.csv
  pwv_climatology: /home/ici/pwv_monthly_means.csv

sky_alignment:
  sun_detection_min_pixels: 100

# ICI processor configuration parameters
# Changing the following values will affect system performance. Consult NWB before adjusting.
# Commenting this block out will cause the ICI to run without processing
processor:

  # Thermal radiance calibrator
  calibrator:
    path: /storage/calibration/post_install.2023-05-20.2023-06-20/camera_calibration.SN104-S0155953.npz

  # Field of view mapping
  sky_map_path: /storage/self_characterization/sky_azimuth_elevation_map.npz

  # Clear sky model
  clear_sky_radiance_model:
    class: IciPwvSurfTempClearSkyRadianceModel

  # Output path

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo      M-A Mark Text
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line M-E Redo      M-6 Copy Text
```

Figure 12. Setting the calibration path and sky map path in the configuration file.

Once the calibration has been updated, restart the image capture and processor service so that it loads the new calibration:

```
sudo systemctl restart nwb-ici-icp.service
```

Finally, sanity check the output products from the instrument to ensure proper operation.

Instrument Self-Characterization Operation

The self-characterization algorithm runs automatically at midnight (UTC) each day. No user interaction with the algorithm is necessary. After initial installation of the instrument, the user will typically need to wait several days for the first characterization to run successfully. The alignment portion of the algorithm waits until it has enough sun detections before it allows the algorithm to proceed. Similarly, the fixed pattern correction portion of the algorithm waits for enough clear sky detections to occur before it allows the algorithm to finish.

Once it successfully completes, the self-characterization service will output the alignment, ground mask, and fixed pattern correction data products. The console output from the self-characterization algorithm can be monitored by following its journal output as follows:

```
sudo journalctl -f -u nwb-ici-self-characterization
```

Example output from the algorithm is shown in Figure 13. This console output can be inspected to determine what is limiting the algorithm from completing.

```
Aug 28 17:39:16 ici-1 python3[9449]: INFO     [18.606 self_characterization.py:128] Starting ICI self-characterization
Aug 28 17:39:16 ici-1 python3[9449]: INFO     [19.144 self_characterization.py:207] Self characterization output path: /storage/self_characterization
Aug 28 17:39:16 ici-1 python3[9449]: INFO     [19.145 self_characterization.py:214] Alignment period: 2023-08-23 17:39:16.490833 - 2023-08-28 17:39:16.490850
Aug 28 17:39:16 ici-1 python3[9449]: INFO     [19.145 self_characterization.py:217] Retrieving sun detections
Aug 28 17:39:42 ici-1 python3[9449]: INFO     [44.305 self_characterization.py:233] Sun detections occur over 2023-08-23 17:40:00 - 2023-08-28 17:39:00
Aug 28 17:39:42 ici-1 python3[9449]: INFO     [44.305 self_characterization.py:235] Attempting to align using original image position and sun detections...
Aug 28 17:40:06 ici-1 python3[9449]: INFO     [68.778 self_characterization.py:238] Sun alignment inliers: 3026  outliers: 0  RMSE (deg): 0.30
Aug 28 17:40:06 ici-1 python3[9449]: INFO     [68.778 self_characterization.py:240] Successfully found alignment with original image position
Aug 28 17:40:06 ici-1 python3[9449]: INFO     [68.779 self_characterization.py:244] Attempting to align using detected image position and sun detections...
Aug 28 17:40:06 ici-1 python3[9449]: INFO     [68.779 self_characterization.py:247] Finding minimum image used for image position detection
Aug 28 17:40:16 ici-1 python3[9449]: INFO     [78.518 self_characterization.py:324] Attempting to find image position...
Aug 28 17:40:18 ici-1 python3[9449]: INFO     [80.277 self_characterization.py:340] FOV center: (264.8, 309.8)  Edge radius: 267.4
Aug 28 17:40:18 ici-1 python3[9449]: INFO     [80.277 self_characterization.py:342] Lens shift. New center - old center: (6.250000, -1.750000)
Aug 28 17:40:19 ici-1 python3[9449]: INFO     [82.106 self_characterization.py:346] Aligning using detected image position and sun detections...
Aug 28 17:40:43 ici-1 python3[9449]: INFO     [105.820 self_characterization.py:350] Sun alignment inliers: 2413  outliers: 613  RMSE (deg): 0.72
Aug 28 17:40:43 ici-1 python3[9449]: INFO     [105.821 self_characterization.py:352] Successfully found alignment with detected image position
Aug 28 17:40:43 ici-1 python3[9449]: INFO     [105.821 self_characterization.py:368] Found best alignment using original image position
Aug 28 17:40:49 ici-1 python3[9449]: INFO     [112.068 self_characterization.py:434] Identifying clear sky images within the period used for ground masking and
fixed pattern correction
Aug 28 17:40:56 ici-1 python3[9449]: INFO     [118.709 self_characterization.py:497] Found 23460 images with clear sky. Using 100 images
Aug 28 17:40:56 ici-1 python3[9449]: INFO     [118.709 self_characterization.py:501] Loading and modeling clear sky images
Aug 28 17:41:34 ici-1 python3[9449]: INFO     [156.926 self_characterization.py:554] Deriving ground mask
Aug 28 17:41:34 ici-1 python3[9449]: INFO     [157.145 self_characterization.py:572] Ground mask regions: 2
Aug 28 17:41:37 ici-1 python3[9449]: INFO     [159.969 self_characterization.py:604] Creating fixed pattern correction
Aug 28 17:41:42 ici-1 python3[9449]: INFO     [164.873 self_characterization.py:655] New calibration with FPC is stored at
/storage/self_characterization/camera_calibration.SN104-S0155953.fpc.npz
Aug 28 17:41:42 ici-1 python3[9449]: INFO     [165.004 self_characterization.py:660] Sky map file can be downloaded from
/storage/self_characterization/sky_azimuth_elevation_map.nc
Aug 28 17:41:42 ici-1 python3[9449]: INFO     [165.004 self_characterization.py:661] Successfully completed self-characterization
```

Figure 13. Example console output from the self-characterization service.

In addition, the self-characterization algorithm can be manually triggered (as opposed to waiting until it autonomously runs at midnight UTC) by executing the following command:

```
sudo systemctl start nwb-ici-self-characterization
```

Once the self-characterization runs successfully, the alignment and ground mask are contained in the newly generated file, */storage/self_characterization/sky_azimuth_elevation_map.nc*. Although the script enforces its own quality controls to ensure that the resulting alignment is acceptable, the *sun-alignment.png* file located in the same folder can be used to verify the accuracy of the alignment. A good alignment has a root mean squared error (RMSE) of less than 0.5 degrees. Figure 14 shows an example of a good alignment and a bad alignment. If you have concerns about the accuracy of an alignment, please contact NWB Sensors for assistance.

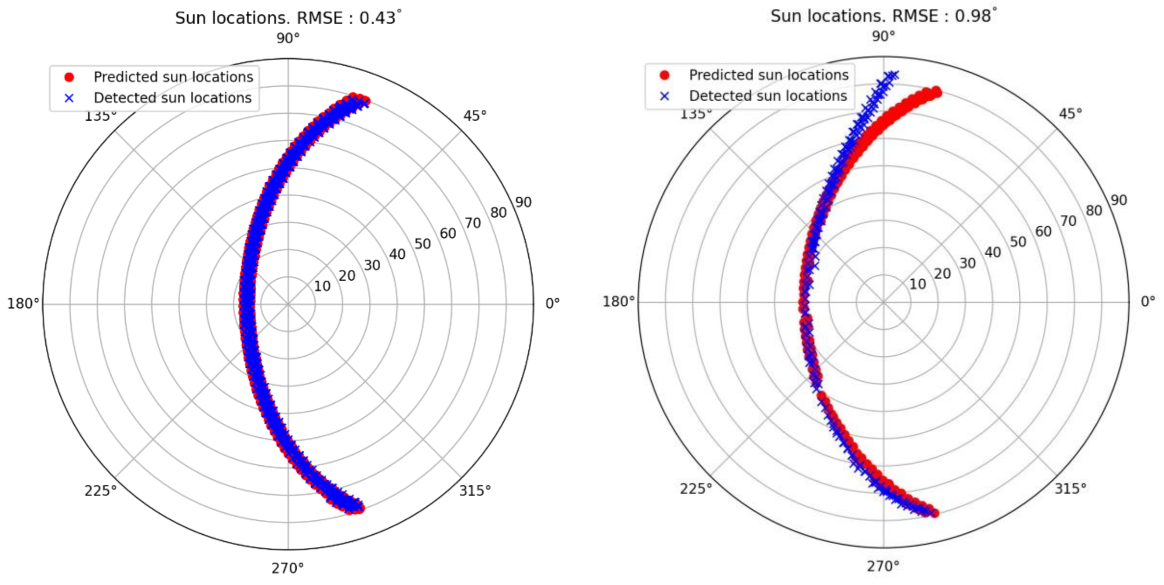


Figure 14. Examples of a good alignment (left) and bad alignment with systematic errors.

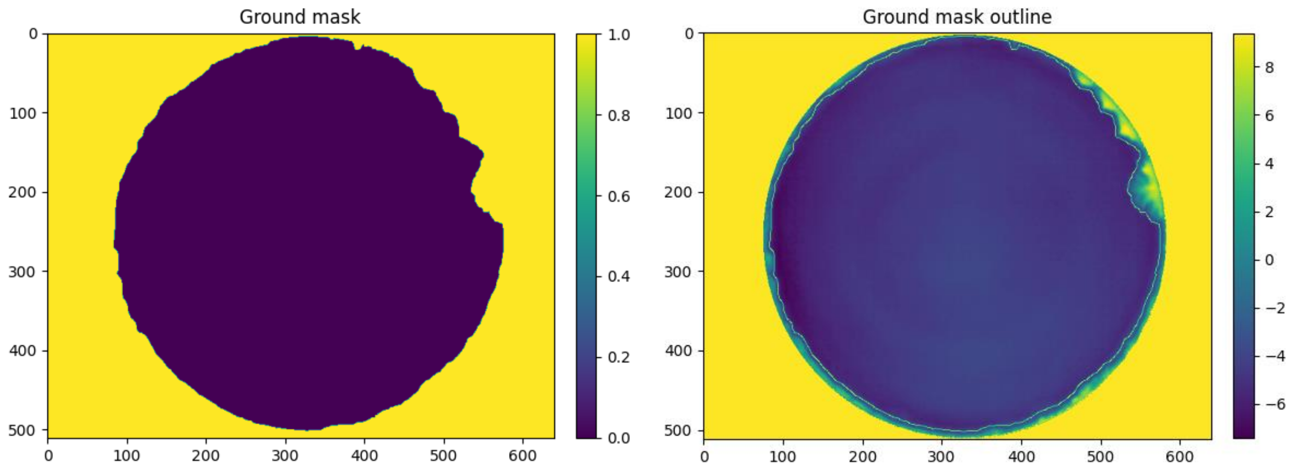


Figure 15. Acceptable ground mask and associated outline.

A rendering of the ground mask can be found at */storage/self_characterization/ground_mask.png*. This should be downloaded from the instrument as well to determine if the ground mask is acceptable. An acceptable ground mask will properly delineate the image into sky and ground regions (see Figure 15).

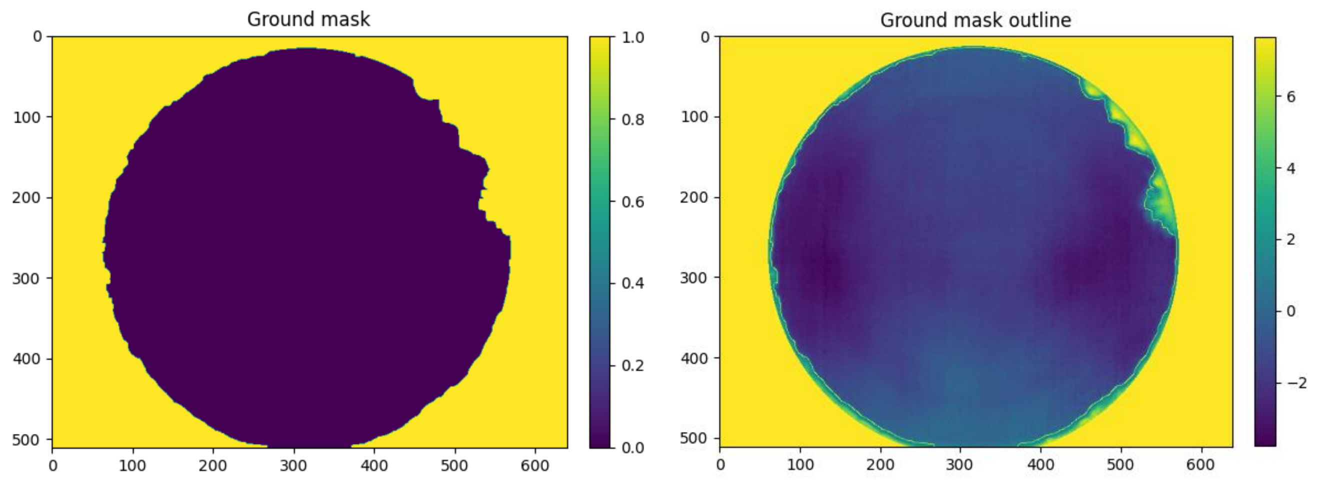


Figure 16. Ground mask which fails to mask enough of the treetops.

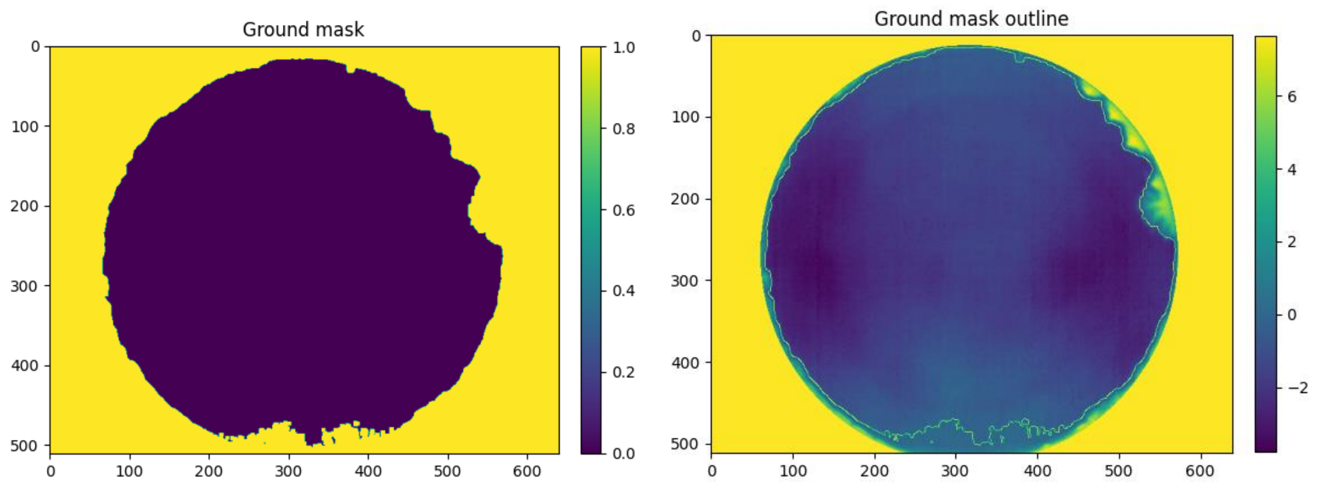


Figure 17. Ground mask which masks sky as ground.

The algorithm masks images areas where the detected clear sky radiances consistently diverge from the clear sky radiance model. Autonomously delineating ground from sky can be impossible in situations where diffuse objects such as trees exist. The algorithm can fail to mask all ground objects (see Figure 16) or can mask portions of the sky as ground (Figure 17). The ground mask is provided as a best effort. If a more accurate mask is needed by the end user, it may be necessary to manually modify the derived mask (using Python or a similar language) to achieve the desired result.

Software Configuration

Operational Parameters

All ICI services share a common YAML configuration file located in the ici_driver python package (see *Filesystem Locations*). Several parameters within the configuration file can be tailored by the operator.

These are listed in Table 9. Root level keys are listed in bold with nested field keys listed below each root key. **Only the parameters listed below are configurable by the operator. Do not change any other configuration parameters without direction from NWB Sensors.**

Key	Description
station	
latitude_deg	Station latitude in decimal degrees where negative is South.
longitude_deg	Station longitude in decimal degrees where negative is West.
altitude_m	Station altitude in meters.
IMPORTANT: The station section of the configuration is optional. If defined, it will override the location measured by the GNSS system. For systems which do not use the GNSS system option, station location information **must** be manually entered into these fields at startup to allow the onboard alignment to act.	
Operation	
verbose	Sets the journal output level: ERROR, WARNING, INFO, or DEBUG Setting this to DEBUG will cause the system journals to fill quickly and rotate more often. DEBUG would only be used when assisted by NWB Sensors in troubleshooting a problem with the system.
frame_interval_sec	Seconds between image captures. Do not set this faster than 15 seconds or the ICP will fail to capture frames at the desired rate and frames will be skipped.
high_wind_threshold_mps	Wind speed at which the hatch will close to protect from blowing debris. By default, this is set to 20.5 m/s (46 mph, 40 knots).
high_wind_reset_sec	Period of time over which no recent wind observation can be above the high wind threshold for the high wind condition to clear and the hatch to reopen.
maximum_weather_latency_hours	Maximum latency of a weather observation before it is regarded as stale.
maximum_pwv_latency_hours	Maximum latency of a PWV observation before it is regarded as stale.
default_air_temp_c	Surface air temperature in Celsius to use as a last resort when weather data are unavailable and climatologies are not set.
default_relative_humidity_pct	Surface relative humidity in percent to use as a last resort when weather data are unavailable.
default_station_pressure_mb	Surface barometric pressure (station pressure) in millibar to use as a last resort when weather data are unavailable.
default_pwv_cm	PWV in centimeters to use as a last resort when PWV data are unavailable and climatologies are not set.
temperature_climatology	Absolute path to temperature climatology file. By default, this is set to /home/ici/surface_temperature_monthly_hourly_means.csv. If this file is not found, the processor assumes no climatology is set. See next section for format of the CSV climatology file.
pwv_climatology	Absolute path to PWV climatology file. By default, this is set to /home/ici/ pwv_monthly_hourly_means.csv. If this file is not found, the processor assumes no climatology is set. See next section for format of the CSV climatology file.
processor	
calibration.path	Path to camera calibration file. This is set by NWB Sensors during system characterization and testing. Post-installation characterization make require updating this as described in the post installation section.

sky_map_path	Path to the file containing the post-installation derived local coordinate frame azimuth and zenith mappings and ground mask. These are used by the processor to accurately model the predicted sky radiance and to mask the cloud products for ground objects. See the post-installation derivation section for more information.
netcdf.prefix	User configurable prefix for output NetCDF file names. By default, all NetCDF output files start with "ici_".
netcdf.save_modeled_clear_sky_radiance	When set to true, an additional parameter, *modeled_clear_sky_radiance_wpm2sr*, is added to the NetCDF output which contains the modeled sky radiance. See *Data Products* section for more information.
netcdf.save_calibrated_radiance	When set to true, an additional parameter, *calibrated_radiance_wpm2sr*, is added to the NetCDF output which contains the unmodified calibrated data. See *Data Products* section for more information.

Table 9. User modifiable configuration parameters

NTP Server Configuration

By default, the ICI uses a pool.ntp.org server. A local NTP server can be configured by setting the *FallbackNTP* server field in /etc/systemd/timesyncd.conf to the IP or hostname of the NTP server.

The following line is an example of this setting:

```
FallbackNTP=10.178.62.5
```

Once set, restart the NTP service by issuing the following command:

```
sudo systemctl restart systemd-timesyncd
```

Filesystem Locations

Default filesystem locations for key components are listed in Table 10. File system locations under the */storage/* parent are located on the onboard solid-state drive. All other locations are located on the Jetson's onboard flash (root partition).

Description	Filesystem location
Main package including telemetry, ICP, storage, clean up, and weather station applications.	/usr/local/lib/python3.6/dist-packages/ici_driver
Client API demos (to be copied to customer's host for usage)	/usr/local/lib/python3.6/dist-packages/ici_driver/client
Web page files	/var/www/ici-web-app
Configuration file	/usr/local/lib/python3.6/dist-packages/ici_driver/config/ici_config.yml
Calibration and processor inputs directory	/storage/calibration/
NetCDF cloud product directory	/storage/output/data/ Data for each day is stored in a subdirectory with the following name: *<yyyy-mm-dd>* where *<yyyy-mm-dd>* is the UTC date of image capture
Temperature climatology	/home/ici/surface_temperature_monthly_hourly_means.csv

PWV climatology	/home/ici/pwv_monthly_means.csv
Self-characterization output	/storage/self_characterization

Table 10. Filesystem Locations. Those which are configurable in the YAML configuration file are identified in the last column.

Updating the System

Configuration Update Procedure

The configuration file is a text file that can be edited in any text editor. Typically, this would be performed using *nano* or *vi* in a SSH console terminal. See Table 10 for the configuration file location. Select parameters can be changed in the configuration file (Table 9). Errors introduced to this file could cause the system to crash, and it should only be edited by qualified personnel. To update the configuration file, use the following procedure:

1. Always create a backup of the operating configuration file.
2. Modify the configuration file in place or replace it.
3. Changes will not take effect until the software is restarted. Reboot the entire system through the web interface or by executing a `sudo reboot` command on the SSH console.

Calibration Update Procedure

The ICI uses a calibration file that is unique to and specifically derived for each camera. It contains the radiometric calibration and field mappings for the lens. This file is located in the calibration folder and is preloaded onto the instrument prior to shipment. It is typically stored at */storage/calibration/ camera_calibration.<camera serial number>.npz* where camera serial number is a combination of the lens and FLIR Boson serial numbers.

At the direction of NWB Sensors, the calibration file may need to be updated if the camera is changed, or calibrations are updated. Do not update them without direction from NWB Sensors. To update the calibration files, perform the following procedure:

1. Connect to the ICI via SSH
2. Zip the contents of the calibration directory to a zip file named *calibration_backup_[date].zip* where [date] is the current date in yyyy-mm-dd.
3. Remove all the files placed in the zip file (all old calibration files)
4. Transfer the provided updated calibration zip file to the ICI via SSH
5. Unzip the provided zip file into the calibration folder.
6. The calibration file listed in the previous section should now exist in this folder.
7. Restart the ICI software via the web page or SSH.

Python Software Updates

The ICI uses Python software for most of its operational code. This code should not be modified or changed without direction from NWB Sensors. If a code update is required, NWB Sensors will provide instructions for updating the ICI code.

Common SSH Commands

The following table provides a quick reference for common commands which are executed on the command line through SSH access.

Task	SSH command
View all system telemetry	`python3 -m ici_driver.command.get_telemetry`
Set the force hatch closed state	`python3 -m ici_driver.command.set_force_hatch_closed 1`
Clear the force hatch closed state	`python3 -m ici_driver.command.set_force_hatch_closed 0`
Monitor the ICP service system log	`sudo journalctl -f -u nwb-ici-icp`
Monitor the telemetry service system log	`sudo journalctl -f -u nwb-ici-telemetry`
Monitor the self-characterization log	`sudo journalctl -f -u nwb-ici-self-characterization`
Monitor system log including all ICI services	`sudo tail -f /var/log/syslog`
Check the ICP service status	`sudo systemctl status nwb-ici-icp`
Check the telemetry service status	`sudo systemctl status nwb-ici-telemetry`
Stop the ICP service	`sudo systemctl stop nwb-ici-icp`
Start the ICP service	`sudo systemctl start nwb-ici-icp`
Stop the telemetry service This will also stop the ICP service since it depends on the telemetry service.	`sudo systemctl stop nwb-ici-telemetry`
Start the telemetry service Stopping and restarting this service is commonly performed when updating code or calibration files.	`sudo systemctl start nwb-ici-telemetry`
Check that time is synchronized by NTP	`sudo timedatectl status`

Follow the NTP service log entries to diagnose issues with NTP time synchronization	`sudo journalctl -f -u systemd-timesyncd`
	OR
	`sudo systemctl status systemd-timesyncd`

Reboot the system Use this as first step to resolve NTP synchronization issues.	`sudo reboot`
Shut down the system System must subsequently be power cycled to be brought to a usable state.	`sudo shutdown 0`