H3D Detector APIs

Version 1.2

H3D, Inc.

October 27, 2022

Contents

Change Log

Version 1.2

- Deprecate the “Auto-Stop” endpoint in the REST API and replace it with the “Fixed” endpoint
- Update “Spectrum” endpoint inputs for format and type
- Add new endpoints to REST API: “SaveState”, “system/Shutdown”,” system/Distance”, “system/Laser”, “system/Livetime”, “Status”, “Unit”)
- Add a note about customization and performance of supplied example code for list-mode stream
- Update N42 example to reflect new label for spectrum as well as temperature info in the characteristics
- Add imaging coordinate system diagrams for H100 and P100 series
- Add list-mode data coordinate drawings for H/P/M series systems.

Overview

The H3D detector platform supports three different user accessible API formats. These include a web REST API, an N42.42 based XML API, and a list-mode data API based on a FlatBuffer serialization schema. Any or all of these formats may be enabled on any H3D product depending on the market and customer needs. This document describes the usage and formats for these APIs.

Web REST API

In addition to the web-based GUI that is hosted on the detector systems, a RESTful web API is also available to control the system and request data from the system. The API exclusively uses POST commands with JSON encoded data for some commands/requests. While this API is designed for programmatic access, testing the API can be made easier using a third-party tool like Postman.

Start

Request to start a new measurement

URL

/api/v1/start

Method

POST

Data Params

JSON encoded body

REQUIRED

```
name: [string] - Name of the measurement you are starting
```

Success Response

On success you should see a 200 HTTP Status and an empty body

Code: 200
Content: <empty>

Error Response

On error you will either see 500 or 403

Code: 403 Forbidden
Content: ACCESS DENIED

OR

Code: 500 Internal Server Error
Content: Incorrect Parameters Provided

Sample Call

```
POST /api/v1/start HTTP/1.1
Host: localhost
Content-Type: application/json

{
    "name": "MyMeasName"
}
```

Notes

None.

Stop

Request to stop the current measurement

URL

/api/v1/stop

Method

POST

Data Params

<NONE>

Success Response

On success you should see a 200 HTTP Status and an empty body

Code: 200
Content: <empty>

Error Response

On error you will either see 500 or 403

Code: 403 Forbidden
Content: ACCESS DENIED

OR

Code: 500 Internal Server Error
Content: Incorrect Parameters Provided

Sample Call

```
POST /api/v1/stop HTTP/1.1
Host: localhost
Content-Type: application/json
```

Notes

None.

Repeat

Request to enable repeat measurements and/or set repeat interval

URL

/api/v1/repeat

Method

POST

Data Params

JSON encoded body

OPTIONAL (One parameter must be present)

```
enable: [boolean] – enable (true) or disable (false) repeat measurements
interval: [integer] – repeat interval in seconds
```

Success Response

On success you should see a 200 HTTP Status and an empty body

Code: `200`
Content: `<empty>`

Error Response

On error you will either see 500 or 403

Code: `403 Forbidden`
Content: `ACCESS DENIED`

OR

Code: `500 Internal Server Error`
Content: `Incorrect Parameters Provided`

Sample Call

```
POST /api/v1/repeat HTTP/1.1
Host: localhost
Content-Type: application/json

{
    "enable": true
    "interval": 30
}
```

Notes

None.

Fixed [Auto-stop – *deprecated*]

Request to enable fixed length measurements and/or set fixed length

URL

/api/v1/fixed
/api/v1/auto-stop [deprecated]

Method

POST

Data Params

JSON encoded body

OPTIONAL (one parameter must be present)

```
enable: [boolean] – enable (true) or disable (false) fixed length measurements
length: [integer] – measurement length in seconds
```

Success Response

On success you should see a 200 HTTP Status and an empty body

Code: `200`
Content: `<empty>`

Error Response

On error you will either see 500 or 403

Code: `403 Forbidden`
Content: `ACCESS DENIED`

OR

Code: `500 Internal Server Error`
Content: `Incorrect Parameters Provided`

Sample Call

```
POST /api/v1/fixed HTTP/1.1
Host: localhost
Content-Type: application/json

{
    "enable": true
    "length": 60

}
```

Notes

None.

Spectrum

Request current spectrum

URL

/api/v1/spectrum

Method

POST

Data Params

JSON encoded body

OPTIONAL

```
format: [string] - (json,spe,txt) format of spectrum to return.
type: [string] - (pur,single,individual) type of spectrum to return.
```

Success Response

On success you should see a 200 HTTP Status and spectral data in the body

Code: 200

Content:

Format: json


```
{
    "channel_width": 1,
    "live_time": 39141.01953125,
    "real_time": 39144.81640625,
    "spectrum": [
        0,
        0,
        1,
        0,
        0,
        1,
        0,
        0,
        0,
        0,
        0,
        3,
        13,
        …
        0,
        0,
        0
    ],
    "type": "pur"
}
```


Format: txt

Raw spectral data (counts/bin)

```
0
0
0
0
1
2
1
1
2
1
1
2
2
1
1
1
4
19
53
…
```

Format: spe

Standard SPE format

```
$SPEC_ID:
H3D_MCAT_20210126-185521_SPE
$SPEC_REM:
DETDESC# Polaris-Q 2 Q0202005Sheila
AP# 2021011518
$DATE_MEA:
01/26/2021 18:55:21
$MEAS_TIM:
2321.18 2340
$RT:
2340.59
$DT:
19406
$DATA:
0 8191
78
0
0
0
…
0
282
```

```
0
$MCA_CAL:
2
0 1
$ENER_FIT:
0 1
```

Error Response

On error you will either see 500 or 403

Code: `403 Forbidden`
Content: `ACCESS DENIED`

OR

Code: `500 Internal Server Error`
Content: `Incorrect Parameters Provided`

Sample Call

```
POST /api/v1/spectrum HTTP/1.1
Host: localhost
Content-Type: application/json

{
    "format": spe
    "type": pur
}
```

Notes

When a measurement is running, a call to this endpoint will return the latest integrated spectrum for the current measurement. Once the measurement stops, this endpoint will return the integrated spectrum from the previous measurement and will not change until a new measurement is started.

There are three different types of spectra that can be requested by changing the “type” input parameter. The “pur” type returns a spectrum of all detector events (interactions within a given readout cycle are summed together). The “individual” type returns a spectrum with each interaction binned separately. The “single” type returns a spectrum with 1-interaction events only.

SaveState

Save the current spectrum and image to a directory inside the measurement data directory

URL

/api/v1/savestate

Method

POST

Data Params

JSON encoded body

OPTIONAL

```
notes: [text] – notes to associate with this save state
```

Success Response

On success you should see a 200 HTTP Status and an empty body

Code: 200
Content: [Save state index]

Error Response

On error you will either see 500 or 403

Code: 403 Forbidden
Content: ACCESS DENIED

OR

Code: 500 Internal Server Error
Content: [Error code]

Sample Call

```
POST /api/v1/savestate HTTP/1.1
Host: localhost
Content-Type: application/json

{
    "notes": “These are measurement notes.”
}
```

Notes

- Save state index is simply a monotonically increasing index for each save state
- Error codes for this call are enumerated here:
    - -11: No current directory
    - -12: Too little time between save states
    - -13: No data drive
    - -14: System is in stopped mode
    - -15: System is in search mode

System/Shutdown

Shutdown computer [and power down systems for soft-off detector models]

URL

/api/v1/system/shutdown

Method

POST

Data Params

<NONE>

Success Response

On success you should see a 200 HTTP Status and an empty body

Code: 200

Content: <empty>

Error Response

On error you will either see 500 or 403

Code: 403 Forbidden

Content: ACCESS DENIED

Sample Call

```
POST /api/v1/system/shutdown HTTP/1.1
Host: localhost
Content-Type: application/json
```

Notes

None.

System/Distance

Set the measurement distance

URL

/api/v1/system/distance

Method

POST

Data Params

JSON encoded body

OPTIONAL (At least one parameter must be present. Distance set is sum of the input parameters.)

```
meters: [float] – distance in meters to add to source to detector distance
feet: [float] – distance in feet to add to source to detector distance
inches: [float] - distance in inches to add to source to detector distance
```

Success Response

On success you should see a 200 HTTP Status and an empty body

Code: 200

Content: <empty>

Error Response

On error you will either see 500 or 403

Code: 403 Forbidden
Content: ACCESS DENIED

OR

Code: 500 Internal Server Error
Content: Incorrect Parameters Provided

OR

Code: 500 Internal Server Error
Content: Meters Value Must Be Non-Negative

OR

Code: 500 Internal Server Error
Content: Feet Value Must Be Non-Negative

OR

Code: 500 Internal Server Error
Content: Inches Value Must Be Non-Negative

Sample Call

```
POST /api/v1/system/distance HTTP/1.1
Host: localhost
Content-Type: application/json

{
    "meters": 2
    "feet": 3.5
    "inches": 6
}
```

Notes

The above sample call would result in a set distance of 3.219m.

System/Laser

Fire the laser rangefinder and return the measured distance in meters.

URL

/api/v1/system/laser

Method

POST

Data Params

<NONE>

Success Response

On success you should see a 200 HTTP Status and JSON encoded distance data in the body

Code: 200
Content:

```
{"distance":0.027000000700354576}
```

Error Response

On error you will either see 500 or 403

Code: 403 Forbidden
Content: ACCESS DENIED

OR

Code: 500 Internal Server Error
Content: No Rangefinder Installed

OR

Code: 500 Internal Server Error
Content: Failed To Read Distance

Sample Call

```
POST /api/v1/system/laser HTTP/1.1
Host: localhost
Content-Type: application/json
```

Notes

None.

System/LiveTime

Enable or disable the use of live time for all of the timers in the interface.

URL

/api/v1/system/livetime

Method

POST

Data Params

JSON encoded body

OPTIONAL (If no parameter is present, the state will toggle.)

```
enable: [boolean] - enable (true) or disable (false) the use of live time
```

Success Response

On success you should see a 200 HTTP Status and an empty body

Code: `200`
Content: `<empty>`

Error Response

On error you will either see 500 or 403

Code: `403 Forbidden`
Content: `ACCESS DENIED`

OR

Code: `500 Internal Server Error`
Content: `Incorrect Parameters Provided`

Sample Call

```
POST /api/v1/system/livetime HTTP/1.1
Host: localhost
Content-Type: application/json

{
    "enable": true
}
```

Notes

None.

Status

Return status information. See *Appendix F – Status Information Definitions* for more information about the output.

URL

/api/v1/status

Method

POST

Data Params

JSON encoded body

OPTIONAL

`format: [string] - (json, text) JSON response by default or just raw text value`

Success Response

On success you should see a 200 HTTP Status and JSON encoded status information

Code: 200

Content:


```
{
    "status/config/CustomImageScale": false,
    "status/config/Distance": 14.499999582767487,
    "status/config/MaskPlugState": false,
    "status/config/NearField": false,
    "status/config/PresetTime": -1,
    "status/config/RepeatTime": -1,
    "status/config/UTCOffset": -4,
    "status/config/UsingLiveTime": false,
    "status/measurement/CountRate": 2,
    "status/measurement/CurrentDirectory": "",
    "status/measurement/DeadTimeFraction": 0.00019999999494757503,
    "status/measurement/DoseRate": 0.0009089509944715246,
    "status/measurement/ElapsedSeconds": "62938",
    "status/measurement/MeasurementMode": 2,
    "status/measurement/PreviousDirectory": "",
    "status/measurement/TotalDose": 0.02141918048437219,
    "status/network/RMSIPAddress": "",
    "status/network/ethernet/Gateway": "192.168.3.254",
    "status/network/ethernet/IPAddress": "192.168.3.4",
    "status/network/ethernet/Netmask": "255.255.255.0",
    "status/state/BatteryLevel": "97",
    "status/state/BiasStatus": 0,
    "status/state/CameraFault": false,
    "status/state/DAQError": 0,
    "status/state/DiskFree": 24280,
    "status/state/DiskFreeFraction": 82,
    "status/state/DiskSpaceError": 0,
    "status/state/MaskFault": false,
    "status/state/NearFieldOpticalTooClose": false,
    "status/state/PowerSource": 0,
    "status/state/RelativeHumidity": 9,
    "status/state/StorageInternal": false,
    "status/state/StorageMounted": true
}
```

Error Response

On error you will either see 500 or 403

Code: `403 Forbidden`
Content: `ACCESS DENIED`

OR

Code: `500 Internal Server Error`
Content: `Incorrect Parameters Provided`

Sample Call

```
POST /api/v1/status HTTP/1.1
Host: localhost
Content-Type: application/json
```

Notes

Each endpoint may be accessed directly (e.g. /api/v1/status/config *or* /api/v1/status/config/utcoffset). Adding an underscore to the URL of an individual endpoint will have the same result as defining the "text" format in the parameters. This feature is especially useful when accessing the status via a GET call in a web browser. The format parameter is only used when accessing an individual status endpoint (e.g. /api/v1/status/measurement/TotalDose or /api/v1/status/config/Distance), not when accessing endpoint groups (e.g. /api/v1/status/config).

Unit

Return unit information. See *Appendix G – Unit Information Definitions* for more information about the output.

URL

/api/v1/unit

Method

POST

Data Params

JSON encoded body

OPTIONAL

```
format: [string] - (json, text) JSON response by default or just raw text value
```

Success Response

On success you should see a 200 HTTP Status and JSON encoded status information

Code: `200`
Content:

```
{
    "unit/Config": "CBRN",
    "unit/CustomSN": "",
    "unit/Model": "P100 3",
    "unit/Name": "Oscar",
    "unit/SN": "P0200006",
    "unit/versions/Patch": "2022083017"
}
```


Error Response

On error you will either see 500 or 403

Code: `403 Forbidden`
Content: `ACCESS DENIED`

OR

Code: `500 Internal Server Error`
Content: `Incorrect Parameters Provided`

Sample Call

```
POST /api/v1/unit HTTP/1.1
Host: localhost
Content-Type: application/json
```

Notes

Each endpoint may be accessed directly (e.g. /api/v1/unit/versions *or* /api/v1/unit/versions/patch). Adding an underscore to the URL of an individual end point will have the same result as defining the “text” format in the parameters. This feature is especially useful when accessing the status via a GET call in a web browser. The format parameter is only used when accessing an individual unit endpoint (e.g. /api/v1/unit/Model), not when accessing endpoint groups (e.g. /api/v1/unit/versions).

N42.42 XML API

The H3D detector platform provides status and result information by serving a stream of N42.42 data over TCP/IP connections. Clients may connect to the detector to receive this N42.42 data as well as control the state of the detector by sending commands through this same TCP/IP connection. The standard N42.42 data contains rad information (spectra, nuclide IDs, etc.) as well as status information of the detector (temperature, battery, errors, etc.). Responses to the commands will be sent between complete N42.42 messages.

Connections

Opening a TCP/IP connection to the detector on port 8082 will start the N42.42 message stream. No logon or commands must be sent to receive the ANSI data. The stream is simply ASCII text data.

N42.42 Message Format

The API uses standard 2012 N42.42 data format. However, some of the fields are used in an unconventional way. This list describes the sections which may be ambiguous.

RadInstrumentCharacteristics Element

This section of the N42.42 message contains basic status information about the detector system.

Battery Charge

The “Battery Charge” characteristic describes the current battery level (when applicable).

Selected Isotope

The “Selected Isotope” characteristic indicates the isotope that is currently selected for image reconstruction. If there are no current isotopes selected, this element will not exist; however, if multiple isotopes are selected, there will be a characteristic element for each selected isotope.

```
<RadInstrumentCharacteristics>
  <Characteristic>
    <CharacteristicName>Battery Charge</CharacteristicName>
    <CharacteristicValue>40</CharacteristicValue>
    <CharacteristicValueUnits>%</CharacteristicValueUnits>
    <CharacteristicValueDataClassCode>decimal</CharacteristicValueDataClassCode>
  </Characteristic>
  <Characteristic>
    <CharacteristicName>Selected Isotope</CharacteristicName>
    <CharacteristicValue>Co-60</CharacteristicValue>
    <CharacteristicValueUnits>unit-less</CharacteristicValueUnits>
    <CharacteristicValueDataClassCode>string</CharacteristicValueDataClassCode>
  </Characteristic>
</RadInstrumentCharacteristics>
```

NuclideAnalysisResults Element

Source Position

The source position element describes the direction of the nuclide detected by the imager in the NuclideAnalysisResults element inside of a particular Nuclide element. A direction vector is defined in

the RelativeLocation element contained in the SourcePosition element. The RelativeLocationAzimuthValue and RelativeLocationInclinationValue elements describe the azimuthal and polar values describing the source direction. If the imaging system is aware of the current GPS/INS information, the direction information is absolute to true north. If there is no GPS/INS information, then the direction vector will be relative to the direction of the detector system and the Geographic Point values will be all zeros. See Appendix D for local imaging coordinates and relative directions.

```
<SourcePosition>
  <Remark>Note that the azimuth below is absolute (0=north, 90=east, etc.).</Remark>
  <RelativeLocation>
    <RelativeLocationAzimuthValue>-125.000000</RelativeLocationAzimuthValue>
    <RelativeLocationInclinationValue>53.000000</RelativeLocationInclinationValue>
    <Origin>
      <GeographicPoint>
        <LatitudeValue>42.209515</LatitudeValue>
        <LongitudeValue>-83.740984</LongitudeValue>
        <ElevationValue>125.000000</ElevationValue>
      </GeographicPoint>
    </Origin>
  </RelativeLocation>
</SourcePosition>
```

Commands

To send a command, the client should compose the command and send it through the same TCP/IP connection that it is using to read the N42.42 messages. For each of the following commands, the detector will respond with an acknowledgement, error message, or a specialized response described in the command specification. The acknowledgement and error message formats are:

```
<?xml version="1.0" encoding="UTF-8"?>
<response type="acknowledge" id=”ID”/>

<?xml version="1.0" encoding="UTF-8"?>
<response type="error" id=”ID”>
    <param name=”message” value=”Some error message here…”/>
</response>
```

Note that for all communication between the server and client, a unique ID is attached as an attribute in the top node for reference when an acknowledgement/response is sent. This ID shall be a monotonically increasing positive integer, e.g., seconds since some epoch.

Request Current Isotope List

This request will prompt the detector to respond with the current list of isotopes in the library.

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<command type="requestCurrentIsotopeList" id="ID"/>
```

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<response type="currentIsotopeList" id="ID">
    <param name="isotope" value="Cs-137"/>
    <param name="isotope" value="K-40"/>
</response>
```

Start Measurement

This command will start a new measurement with the optional specified name. A fixed time measurement can also be specified with the optional arguments.

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<command type="newMeasurement" id="ID"/>
```

OR

```
<?xml version="1.0" encoding="UTF-8"?>
<command type="newMeasurement" id="ID">
    <param name="name" value="MeasurementName"/> (optional)
    <param name="fixedTime" value="300"/> (optional)
</command>
```

* NOTE: The name and fixedTime parameters are optional. If the name is not specified, "AutoName" will be used. If the fixedTime is not specified, the measurement will run indefinitely. The measurement time is specified in seconds.

Stop Command

This command will stop the current measurement. If the measurement is already stopped, nothing will happen.

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<command type="stopMeasurement" id="ID"/>
```

Select Isotopes

This command will set currently selected isotope(s). The radiation images corresponding to these selected isotopes are included in the reconstructed image which is included in the N42.42 stream.

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<command type="selectIsotope" id="ID">
    <param name="isotope" value="Cs-137"/>
    <param name="isotope" value="K-40"/>
…
</command>
```

Snap New Optical

This command will take a new photo of the scene around the detector. It will show up in the N42.42 stream.

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<command type="newOptical" id="ID"/>
```

List-Mode API

On detector systems with this interface enabled, the list-mode API provides a steady data stream of list-mode gamma-ray evet information along with time sync data and GPS information (if available). No requests or commands are currently accepted through this protocol.

Connections

The data stream is serialized using a FlatBuffer format defined by the schema file in Appendix B. The data stream is hosted using a TCP/IP connection on port 11503. When a client connects to port 11503, binary data will start streaming through the connection with the following format:

```
PacketSize1 (uint32)
DataPayload1 (PacketSize1 * 1 byte)
PacketSize2 (uint32)
DataPayload2 (PacketSize2 * 1 byte)
...
```

PacketSize: the size of the payload data package in bytes (in little-endian format)
DataPayload: the serialized FlatBuffer binary object containing the list-mode data

Interpreting Data Payload

The H3D list-mode data stream is serialized using the open source FlatBuffer library. Please refer to the FlatBuffer website for documentation on how to use the library to parse the data stream for a variety of languages. Example code for C++ and Python can be found in Appendix C. Keep in mind that these examples are just intended to introduce the user to interpreting the data stream. For production software, the user is responsible for customizations and optimizations. For example, the Python example will not work well when count rates are elevated above a few thousand counts per second. This limitation is of course affected by client side computational power.

List-Mode Data Structure

The list-mode data stream contains several different types of data. These data range from gamma-ray interaction data to coded-aperture mask position information, to timing data. Each data payload consists of an array of each type of data. Note that not all types of data will be present in every payload.

Gamma Event

The **GammaEvent** data type contains information describing the gamma-ray interaction data reconstructed from the detector signals during an individual readout cycle. In this structure, an *interaction* is defined as a single instance of a gamma-ray interacting with the detector material (e.g., Compton scatter, photoelectric effect, pair production, etc.) which is described by the x, y, z coordinates of the location of the interaction as well as the energy deposited during that interaction. An *event* consists of the collection of *interactions* that happened during a single detector readout cycle. The *chip* number describes which detector crystal index a particular interaction occurred. This information can typically be inferred from the x, y, z position information, but it is included as a convenience.

In most low-count-rate scenarios (on the order of a thousand counts per second or less), it is likely that all of the interactions during a given *event* (readout cycle) are the result of a single incident photon. However, since the photon(s) are moving at the speed of light, the sensor cannot determine the true

sequence of the *interactions* in a given event. Thus, if there are multiple *interactions* in a given *event*, the *interactions* are arbitrarily ordered in the data structure. It is left to the user to analyze the *interactions* to determine the true (or most likely) sequence. In high-count-rate scenarios (on the order of 10s of thousands of counts per second) it is much more likely that chance coincidence will occur; in which case, the *interactions* in a given event are the result of multiple photons not a single incident photon. This of course is also possible, but less likely, in low-count-rate scenarios as well.

In addition to the *interaction* data, the **`GammaEvent`** type contains the live time associated with a given *event* (readout cycle) and a timestamp. The *livetime* can be interpreted as the time between the end of the previous readout cycle and the beginning of the readout cycle associated with the current *event*. The *timestamp* is related to the clock of the FPGA in the detector and can be correlated to wall time via the **`ClockEvent`** type.

Mask Event

In detector systems that have automated spinning mask/antimask coded-aperture mechanisms (i.e., H420 models) the list-mode data will also include **`MaskEvents`**. These events contain information about the state of the mask at the instant of the reported timestamp.

The data that are reported in this type include: *rotating*, *quadrant*, *realtime*, *livetime*, and *timestamp*. The *rotating* value is true if the mask is currently spinning. The *quadrant* value describes which orientation in which the mask is currently positioned (not valid if *rotating* is true):

Position Index	**0**	**1**	**2**	**3**	**4**
Mask Orientation	Mask	Antimask	Mask	Antimask	Unknown/Err

The *livetime* value describes the total accumulated **`GammaEvent`** livetime from the previous **`MaskEvent`** to the current **`MaskEvent`**. The *timestamp* is the same as described in the **`GammaEvent`** description and is based off of the same clock (meaning they are directly comparable).

Clock Event

`ClockEvents` are periodically included to relate the FPGA *timestamp* data to the wall time currently set on the detector system in Unix time. Again, the *timestamp* value is the same as the other two data types. The *clockseconds* value describes the full integral seconds since January 1, 1970 at 12:00:00 AM. The *clocknanoseconds* represents the fraction of a second past the integral second represented by *clockseconds* in units of nanoseconds.

Sync Event

`SyncEvents` are used to provide synchronization data for two or more independent-readout systems linked together with a hardware sync pulse. The user can match the *index* values across the data streams from the independent systems and correct for the offset between the different *timestamp* values of each system which may drift slightly over time.

Appendix A – Example N24.42 data stream

```
<?xml version="1.0" encoding="UTF-8"?>
<RadInstrumentData xmlns="http://physics.nist.gov/N42/2011/N42">
    <RadInstrumentInformation id="RadInstrumentInformation-1">
        <RadInstrumentManufacturerName>H3D, Inc.</RadInstrumentManufacturerName>
        <RadInstrumentIdentifier>Q0202005</RadInstrumentIdentifier>
        <RadInstrumentModelName>Polaris-Q 2</RadInstrumentModelName>
        <RadInstrumentClassCode>Radionuclide Identifier</RadInstrumentClassCode>
        <RadInstrumentVersion>
            <RadInstrumentComponentName>Software</RadInstrumentComponentName>
            <RadInstrumentComponentVersion>Web: V2.9.1W-0-gd05a1f7 IMG: V2.9.1I-0-g05bdab9, V2.9.1I-0-g0cc3749,
V2.9.1I-0-gb6b3cfd, V2.9.1I-0-g1ecc488 DAQ: V3.3.1Q-0-g8683308, V3.3.1Q-0-
g9498660</RadInstrumentComponentVersion>
        </RadInstrumentVersion>
    </RadInstrumentInformation>
    <RadDetectorInformation id="Gamma-0">
        <RadDetectorCategoryCode>Gamma</RadDetectorCategoryCode>
        <RadDetectorKindCode>CZT</RadDetectorKindCode>
    </RadDetectorInformation>
    <EnergyCalibration id="EnergyCalibration-0">
        <CoefficientValues>0 1.000000 0</CoefficientValues>
    </EnergyCalibration>
    <RadMeasurement id="RadMeasurement-1">
        <MeasurementClassCode>Foreground</MeasurementClassCode>
        <StartDateTime>2020-10-13T16:39:48-04:00</StartDateTime>
        <RealTimeDuration>PT41.715000S</RealTimeDuration>
        <Spectrum id="RadMeasurement-1-Spectrum-1-PUR" radDetectorInformationReference="Gamma-0"
energyCalibrationReference="EnergyCalibration-0">
            <LiveTimeDuration>PT39.537201S</LiveTimeDuration>
```

```
            <ChannelData compressionCode="CountedZeroes">0 39 1 0 1 3 2 7 6 8 11 11 11 12 14 8 11 5 11 15 11 12
11 10 11 13 6 8 10 8 16 13 13 12 11 9 13 13 13 19 11 17 14 8 9 10 12 13 9 8 10 12 10 12 10 7 11 11 12 14 15 14
13 14 16 7 8 6 9 10 14 13 11 11 12 11 12 18 9 13 15 8 8 13 10 13 12 11 16 15 9 7 11 15 12 8 10 11 8 14 16 10 12
9 17 10 15 10 10 9 19 12 7 10 13 10 14 17 12 14 11 11 10 12 13 12 10 10 8 8 10 12 7 12 13 10 13 10 7 14 14 14 9
12 9 6 8 12 15 12 9 12 8 4 5 10 14 13 13 8 11 9 8 12 6 9 12 11 7 9 11 12 9 5 13 10 11 10 10 12 14 6 11 9 8 11 11
11 12 10 11 12 9 10 12 14 11 6 9 8 6 9 10 5 6 7 4 2 7 10 9 10 12 9 9 6 3 5 6 11 9 7 9 11 8 6 12 3 6 10 8 6 5 7 5
8 6 4 7 6 9 10 10 10 7 6 5 14 10 7 8 6 8 8 6 5 8 7 4 8 4 7 8 8 9 5 4 5 10 5 6 4 5 8 8 10 7 9 7 8 5 6 7 6 8 5 2 4
5 4 6 3 6 5 2 5 4 8 5 6 5 9 4 6 8 7 4 1 5 4 5 9 6 4 3 5 8 9 5 2 5 5 2 4 3 3 6 4 4 5 6 5 4 5 2 6 5 3 2 4 4 3 3 5
4 2 2 7 3 2 4 4 5 5 5 3 5 3 1 4 5 2 3 3 3 2 5 6 4 4 3 1 6 8 6 6 3 2 5 3 1 4 5 8 2 1 2 2 4 1 3 2 2 6 5 5 5 3 3 6
4 2 3 2 3 4 4 8 6 4 4 5 2 3 3 2 1 1 3 5 4 3 5 2 7 2 4 3 2 3 4 2 3 4 4 4 2 4 4 2 5 1 2 1 6 7 4 3 4 4 4 3 3 0 1 2
3 4 6 4 1 2 3 2 4 4 2 1 3 2 4 3 2 3 2 2 4 6 7 2 7 6 4 3 4 3 2 3 2 2 3 4 2 8 5 6 7 2 4 3 3 5 4 2 2 3 7 5 5 2 3 3
2 6 5 4 4 3 2 3 4 6 3 5 4 2 4 2 2 3 5 4 5 3 4 4 2 4 4 2 1 2 3 3 1 2 4 6 6 5 3 3 5 3 2 4 3 3 3 3 2 2 2 3 2 4 6 4
1 0 1 3 2 3 3 2 4 5 2 2 3 5 3 3 2 2 4 2 4 3 4 2 2 1 1 4 1 3 3 3 1 2 1 2 3 3 2 3 0 1 2 4 2 1 4 1 2 2 2 3 3 2 3 3
4 1 2 6 4 2 2 1 3 2 3 6 2 1 3 3 6 5 4 4 3 2 4 1 2 3 2 3 1 4 6 5 5 4 3 4 2 4 2 3 1 5 3 3 3 3 1 3 2 4 5 5 2 0 1 1
1 2 1 4 4 3 3 3 2 2 2 3 1 2 3 9 4 5 3 3 0 1 2 4 2 3 6 4 2 2 3 6 6 4 3 3 5 4 3 1 2 5 1 3 8 3 2 1 0 1 2 2 3 3 3 1
0 1 2 1 4 3 4 4 4 6 4 2 2 2 2 2 3 3 4 4 2 3 3 0 1 3 4 3 3 2 2 1 2 2 4 2 1 1 4 4 2 2 3 3 1 1 5 1 3 4 2 3 1 2 4 3
2 1 3 4 2 1 1 1 3 4 3 4 3 0 1 1 2 3 3 4 3 3 1 2 3 1 3 3 1 1 2 5 1 2 3 3 2 5 4 5 2 5 3 3 5 2 2 5 5 5 2 3 2 5 5 4
4 5 3 2 3 1 5 5 4 4 1 1 5 4 4 4 3 3 3 3 1 3 2 1 3 2 2 5 4 0 1 1 3 4 3 2 2 2 3 3 3 3 5 5 3 3 4 1 4 3 1 3 3 3 5 3
2 4 2 3 2 2 0 1 1 1 4 3 2 3 2 3 3 2 5 6 7 6 4 2 0 1 3 3 3 0 2 2 4 2 1 1 3 4 2 1 2 1 2 2 1 2 2 2 2 2 4 2 0 1 1 3
2 2 2 0 1 2 3 1 1 3 0 1 1 1 1 1 1 2 0 1 2 3 2 4 3 1 3 1 1 3 4 3 5 2 2 3 2 0 1 2 3 1 2 4 1 1 2 1 3 5 2 2 0 1 2 4
3 2 5 2 2 3 1 1 1 3 1 0 1 2 2 2 2 2 3 2 0 1 1 3 4 4 2 5 2 0 1 2 1 2 1 3 2 3 2 1 2 2 2 2 4 1 3 3 2 1 1 1 1 2 2 1
2 3 3 4 4 1 1 2 2 1 2 2 2 2 1 3 3 1 1 0 1 4 5 4 4 5 3 3 2 2 0 1 2 2 1 2 3 2 3 1 2 2 1 1 2 2 2 2 2 1 1 2 3 3 2 3
4 3 0 1 3 4 1 1 1 2 2 3 2 6 4 6 3 6 5 4 4 4 3 4 4 8 3 10 9 5 9 8 6 8 7 5 7 2 5 5 2 1 1 2 4 2 2 2 0 1 1 3 1 2 3 2
1 0 2 1 0 1 1 0 3 1 2 1 1 1 1 1 0 1 1 0 2 1 0 1 2 2 2 0 3 2 2 0 2 1 1 0 5 1 1 0 1 1 0 6 1 0 1 1 0 1 2 1 0 2 2 0
1 1 1 1 1 0 1 1 0 5 1 0 3 1 0 1 1 0 1 1 1 2 2 1 0 1 1 1 0 1 1 0 3 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 2 1
2 4 2 2 4 3 1 0 2 1 1 1 5 4 4 3 3 5 5 6 5 3 6 5 3 4 4 5 5 3 1 3 1 2 3 3 4 2 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 5 1
0 1 1 0 3 1 0 4 1 0 26 1 1 2 0 38 1 0 10 1 0 10 1 0 25 1 0 25 1 0 81 </ChannelData>
        </Spectrum>
        <ExposureRate id="RadMeasurement-1-ExposureRate-1" radDetectorInformationReference="Gamma-0">
            <CountData units="mR/h">0.071272</CountData>
        </ExposureRate>
        <RadInstrumentState radInstrumentInformationReference="RadInstrumentInformation-1">
```

```
            <RadInstrumentModeCode>Long Dwell</RadInstrumentModeCode>
            <StateVector>
                <RelativeLocation>
                    <DistanceValue type="Unknown">0.314</DistanceValue>
                </RelativeLocation>
            </StateVector>
            <RadInstrumentCharacteristics>
                <Characteristic valueOutOfLimits="false">
                    <CharacteristicName>Battery Charge</CharacteristicName>
                    <CharacteristicValue>98</CharacteristicValue>
                    <CharacteristicValueUnits>%</CharacteristicValueUnits>
                    <CharacteristicValueDataClassCode>decimal</CharacteristicValueDataClassCode>
                </Characteristic>
                <Characteristic valueOutOfLimits="false">
                    <CharacteristicName>Detector Temperature</CharacteristicName>
                    <CharacteristicValue>0.400000</CharacteristicValue>
                    <CharacteristicValueUnits>° C</CharacteristicValueUnits>
                    <CharacteristicValueDataClassCode>decimal</CharacteristicValueDataClassCode>
                </Characteristic>
                <Characteristic valueOutOfLimits="false">
                    <CharacteristicName>Selected Isotope</CharacteristicName>
                    <CharacteristicValue>Co-60</CharacteristicValue>
                    <CharacteristicValueUnits>unit-less</CharacteristicValueUnits>
                    <CharacteristicValueDataClassCode>NonBlankString</CharacteristicValueDataClassCode>
                </Characteristic>
            </RadInstrumentCharacteristics>
        </RadInstrumentState>
    </RadMeasurement>
    <AnalysisResults radMeasurementReferences="RadMeasurement-1">
        <NuclideAnalysisResults>
            <Nuclide>
                <NuclideName>Co-60</NuclideName>
```

```
                <NuclideIdentifiedIndicator>true</NuclideIdentifiedIndicator>
                <NuclideIDConfidenceValue>95</NuclideIDConfidenceValue>
                <SourcePosition>
                    <Remark>Note that the azimuth below is absolute (0=north, 90=east, etc.).</Remark>
                    <RelativeLocation>
                        <RelativeLocationAzimuthValue>49.000000</RelativeLocationAzimuthValue>
                        <RelativeLocationInclinationValue>5.000000</RelativeLocationInclinationValue>
                        <Origin>
                            <GeographicPoint>
                                <LatitudeValue>0.000000</LatitudeValue>
                                <LongitudeValue>0.000000</LongitudeValue>
                                <ElevationValue>0.000000</ElevationValue>
                            </GeographicPoint>
                        </Origin>
                    </RelativeLocation>
                </SourcePosition>
            </Nuclide>
        </NuclideAnalysisResults>
    </AnalysisResults>
<MultimediaData>
        <MultimediaDataDescription>Front Optical Image</MultimediaDataDescription>
        <MultimediaDataMIMEKind>jpeg</MultimediaDataMIMEKind>
        <BinaryBase64Object>/9j/4AAQSkZJRgABAQAA … ooooA//2Q==</BinaryBase64Object>
    </MultimediaData>
    <MultimediaData>
        <MultimediaDataDescription>Radiation Image</MultimediaDataDescription>
        <MultimediaDataMIMEKind>png</MultimediaDataMIMEKind>
        <BinaryBase64Object>iVBORw0KGgoAAAA … ABJRU5ErkJggg==</BinaryBase64Object>
    </MultimediaData>
</RadInstrumentData>
```

Appendix B – H3D List-Mode Data FlatBuffer Schema

```
// H3D IDL file

namespace H3DData;

// Structs \\

struct Interaction {
  energy :uint;                // Energy in eV
  x      :short;               // X position in 10s of microns
  y      :short;               // Y position in 10s of microns
  z      :short;               // Z position in 10s of microns
  chip   :ubyte;               // Detector index in which this event occurred
  extra  :ubyte;               // Reserved for future use
  }


// Tables \\

table SyncEvent {
  index      :ulong;           // Synchronization pulse index
  timestamp  :ulong;           // Synchronization timestamp in 10s of nanoseconds (time since system boot)
}

table ClockEvent {
  clockseconds      :uint;     // Number of full seconds since epoch
  clocknanoseconds  :uint;     // Nanoseconds since last full second since epoch
  timestamp         :ulong;    // FPGA timestamp in 10s of nanoseconds (time since system boot)
}
```

```
table MaskEvent {
  rotating     :bool;          // True if mask is currently rotating
  quadrant     :ubyte;         // Current index of mask quadrant [0,1,2,3] (4 is error state)
  realtime     :uint;          // Real time that has elapsed in current mask position in milliseconds
  livetime     :uint;          // Live time that has elapsed in current mask position in milliseconds
  timestamp    :ulong;         // FPGA timestamp in 10s of nanoseconds (time since system boot)
}

table GammaEvent {
  interactions :[Interaction]; // Vector of interactions
  livetime     :uint;          // Live time associated with this event in 10s nanoseconds
  timestamp    :ulong;         // FPGA timestamp in 10s of nanoseconds (time since system boot)
}

table H3DPacket {
  gammaevents :[GammaEvent];
  clockevents :[ClockEvent];
  syncevents  :[SyncEvent];
  maskevents  :[MaskEvent];
}


root_type H3DPacket;
```

Appendix C – Example Code for Parsing List-Mode Data Stream

C++ Example

```
/*
Authors: Hao Yang and Jason Jaworski
Date: 01/25/2021
Organization: H3D
listModeParser.cpp -- a stream socket client demo

This example c++ function reads data from the list-mode stream published by an H3D detector system.
The function depends on standard c++ libraries and the Google Flatbuffers library.
The user is responsible for generating the H3DData_generated.h file using the flatc compiler.

*/

#include <stdio.h>
#include <stdlib.h>
#include <ctime>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <arpa/inet.h>

#include "H3DData_generated.h"
#include <vector>

#define PORT "11503"

//Set IPv4 address of the detector publishing the data
```

```
#define IP   "192.168.5.10"

int main(int argc, char *argv[])
{

    // Create socket connection ------------------------------------------------------------------
    int sockfd;
    struct addrinfo hints, *servinfo;

    memset(&hints, 0, sizeof hints);
    hints.ai_family = AF_UNSPEC;
    hints.ai_socktype = SOCK_STREAM;

    getaddrinfo(IP, PORT, &hints, &servinfo);

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (connect(sockfd, servinfo->ai_addr, servinfo->ai_addrlen) < 0)
    {
        printf("Connection failed.\n");
        return 1;
    }

    freeaddrinfo(servinfo); // all done with this structure


    // Read data from socket and parse -----------------------------------------------------------
    int nBufferSize(0);
    std::vector <char> Buffer;

    while(1)
    {
        int num2recv(0);
        recv(sockfd, &nBufferSize, sizeof(nBufferSize), 0);
```

```
        Buffer.resize(nBufferSize);
        char *p(Buffer.data());

        while(num2recv != nBufferSize) num2recv += recv(sockfd, p+num2recv, nBufferSize-num2recv, 0);

        auto H3DDataPacket = H3DData::GetH3DPacket(p);
        auto events = H3DDataPacket->gammaevents();
        if (events)
        {
            for(unsigned i=0; i<events->size(); i++)
            {
                auto interactions = events->Get(i)->interactions();

                for(unsigned j=0; j<interactions->size(); j++)
                {
                    auto x= interactions->Get(j)->x();
                    auto y= interactions->Get(j)->y();
                    auto z= interactions->Get(j)->z();
                    auto energy = interactions->Get(j)->energy();
                    auto chip = interactions->Get(j)->chip();
                    //printf("%d %d %d\n", x, y, z);
                }

                auto timestamp = events->Get(i)->timestamp();
                auto livetime = events->Get(i)->livetime();

                //printf(%llu %d\n", timestamp, livetime);
            }
        }

        auto syncevents = H3DDataPacket->syncevents();
        if (syncevents)
```

```
        {
            for(unsigned i=0; i<syncevents->size(); i++)
            {
                auto index = syncevents->Get(i)->index();
                auto timestamp = syncevents->Get(i)->timestamp();
            }
        }

        auto clockevents = H3DDataPacket->clockevents();
        if (clockevents)
        {
            for(unsigned i=0; i<clockevents->size(); i++)
            {
                auto clockseconds = clockevents->Get(i)->clockseconds();
                auto clocknanoseconds = clockevents->Get(i)->clocknanoseconds();
                auto timestamp = clockevents->Get(i)->timestamp();
                //printf("%d %d %llu\n", clockseconds, clocknanoseconds, timestamp);
            }
        }

        auto maskevents = H3DDataPacket->maskevents();
        if (maskevents)
        {
            for(unsigned i=0; i<maskevents->size(); i++)
            {
                auto rotating = maskevents->Get(i)->rotating();
                auto quadrant = maskevents->Get(i)->quadrant();
                auto realtime = maskevents->Get(i)->realtime();
                auto livetime = maskevents->Get(i)->livetime();
                auto timestamp = maskevents->Get(i)->timestamp();
                //printf("%d\n", (int)rotating);
            }
```

```
        }
    }

    return 0;
}
```

Python Example

```python
# Author: Jason Jaworski
# Date: 01/25/2021
# Organization: H3D
# listModeParser.py -- a stream socket client demo

# This example python script reads data from the list-mode stream published by an H3D detector system.
# The script depends on the Google Flatbuffers library.
# The user is responsible for generating the H3DData python files using the flatc compiler.
import flatbuffers
import H3DData.ClockEvent
import H3DData.GammaEvent
import H3DData.H3DPacket
import H3DData.Interaction
import H3DData.MaskEvent
import H3DData.SyncEvent

import socket

# Set IPv4 address of the detector publishing the data
HOST = '192.168.5.10'

PORT = 11503
ADDR = (HOST,PORT)

# Create socket connection ------------------------------------------------------
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(ADDR)

# Read data from socket and parse -----------------------------------------------
while True:
    packetSize = int.from_bytes(client.recv(4), byteorder='little', signed=True)
```

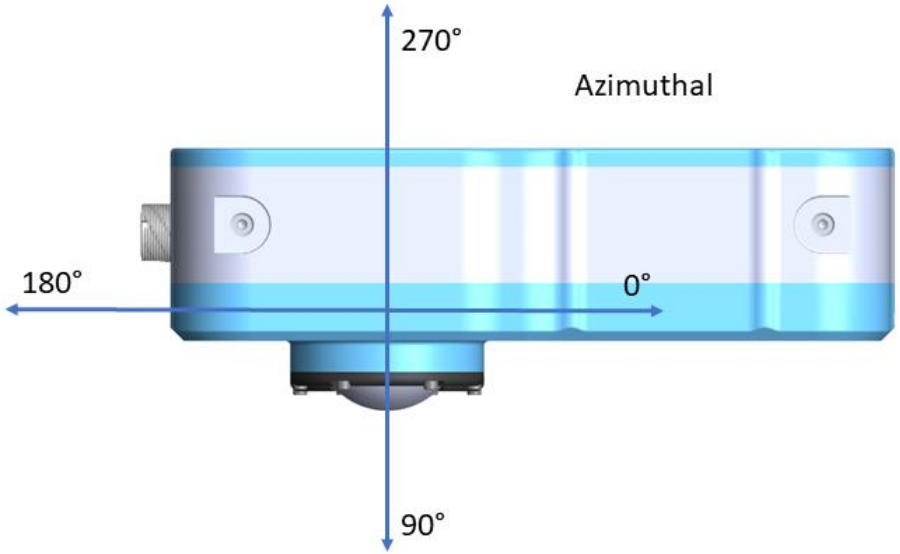
```
    buffer = b""
    while len(buffer) < packetSize:
        buffer += client.recv(packetSize - len(buffer))
    packetIn = H3DData.H3DPacket.H3DPacket.GetRootAsH3DPacket(buffer, 0)
    print("Gamma events:", packetIn.GammaeventsLength())
    if (packetIn.GammaeventsLength() > 0):
        print("Num interactions:", packetIn.Gammaevents(0).InteractionsLength())
        print("Livetime:", packetIn.Gammaevents(0).Livetime())
        print("Timestamp:", packetIn.Gammaevents(0).Timestamp())
        print("X:", packetIn.Gammaevents(0).Interactions(0).X())
        print("Y:", packetIn.Gammaevents(0).Interactions(0).Y())
        print("Z:", packetIn.Gammaevents(0).Interactions(0).Z())
        print("Chip:", packetIn.Gammaevents(0).Interactions(0).Chip())
        print("Energy:", packetIn.Gammaevents(0).Interactions(0).Energy())

    print("Clock events:", packetIn.ClockeventsLength())
    if (packetIn.ClockeventsLength() > 0) :
        print("Clock Seconds:", packetIn.Clockevents(0).Clockseconds())
        print("Clock Nanoseconds:", packetIn.Clockevents(0).Clocknanoseconds())
        print("Timestamp:", packetIn.Clockevents(0).Timestamp())

    print("Sync events: ", packetIn.SynceventsLength())
    if packetIn.SynceventsLength() > 0:
        print("Index:", packetIn.Syncevents(0).Index())
        print("Timestamp:", packetIn.Syncevents(0).Timestamp())

    print("Mask events: ", packetIn.MaskeventsLength())
    if packetIn.MaskeventsLength() > 0:
        print("Rotating:", packetIn.Maskevents(0).Rotating())
        print("Quadrant:", packetIn.Maskevents(0).Quadrant())
        print("Realtime:", packetIn.Maskevents(0).Realtime())
        print("Livetime:", packetIn.Maskevents(0).Livetime())
```

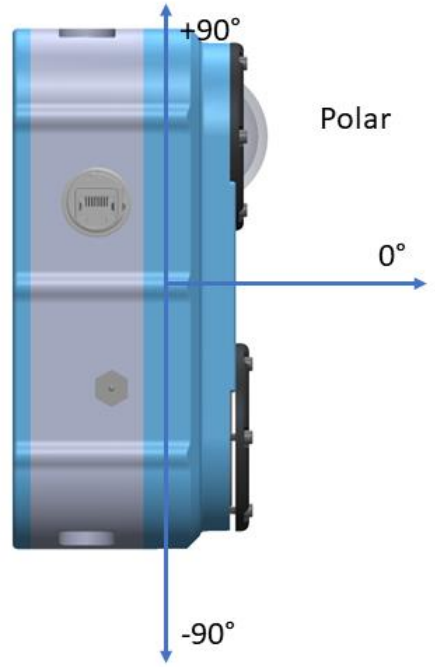
```
        print("Timestamp:", packetIn.Maskevents(0).Timestamp())
```

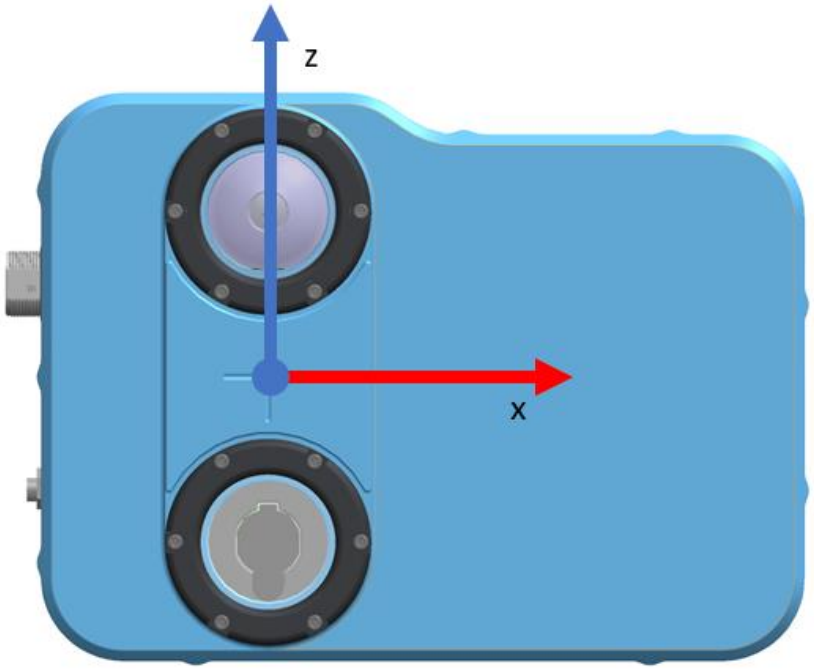
Appendix D – System Local Imaging Coordinate System and Relative Directions

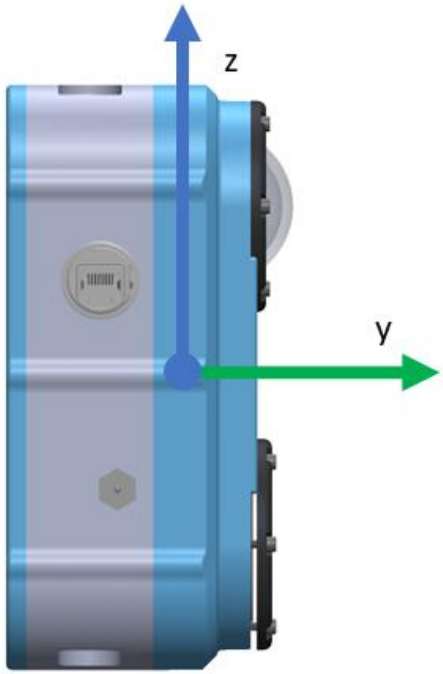
Note: This section describes the coordinate system of reconstructed images (like those found in the N42.42 stream) NOT the coordinate system for the raw list-mode x, y, z data. For list-mode coordinates see Appendix E – List-Mode Data Coordinate Systems.

H100 Series

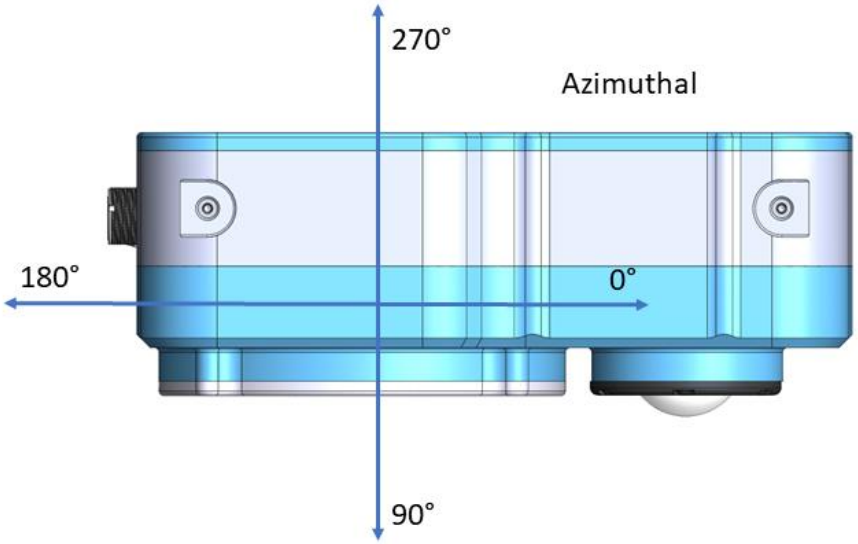


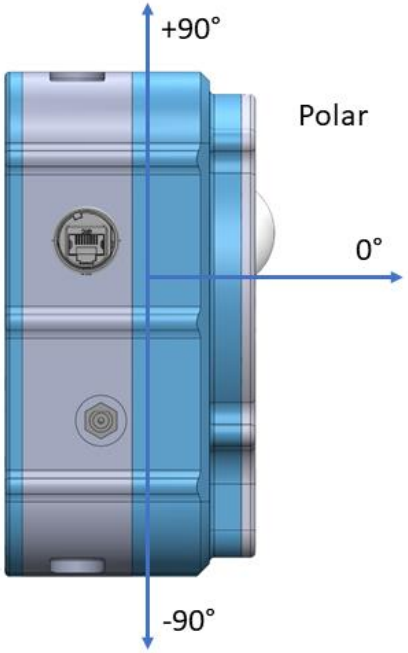


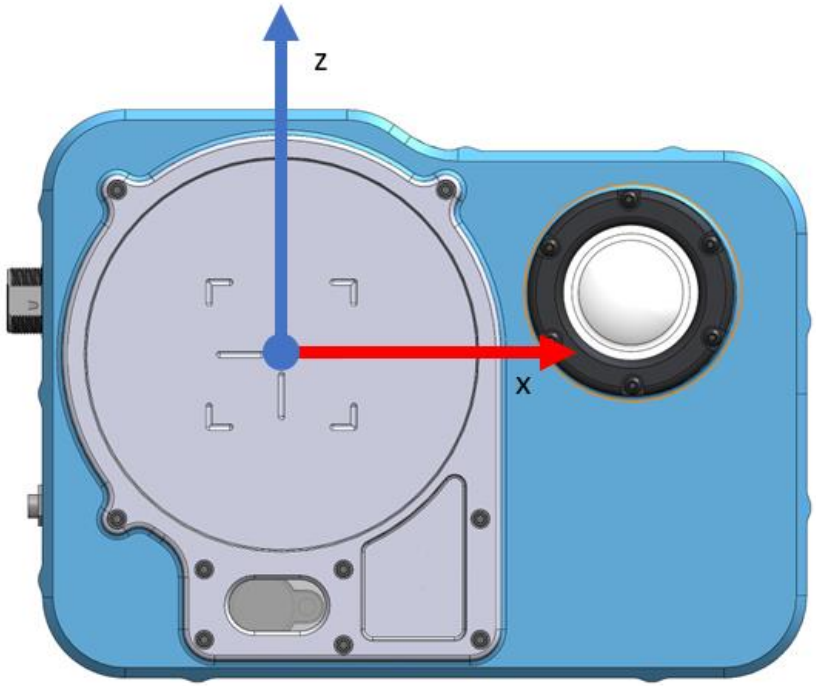

z
x

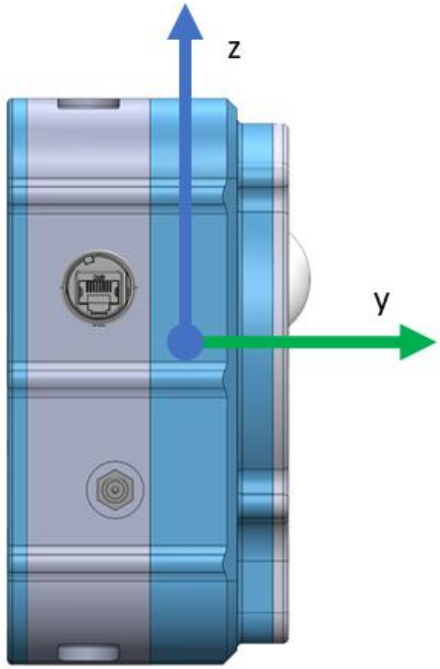

z
y

H400 Series

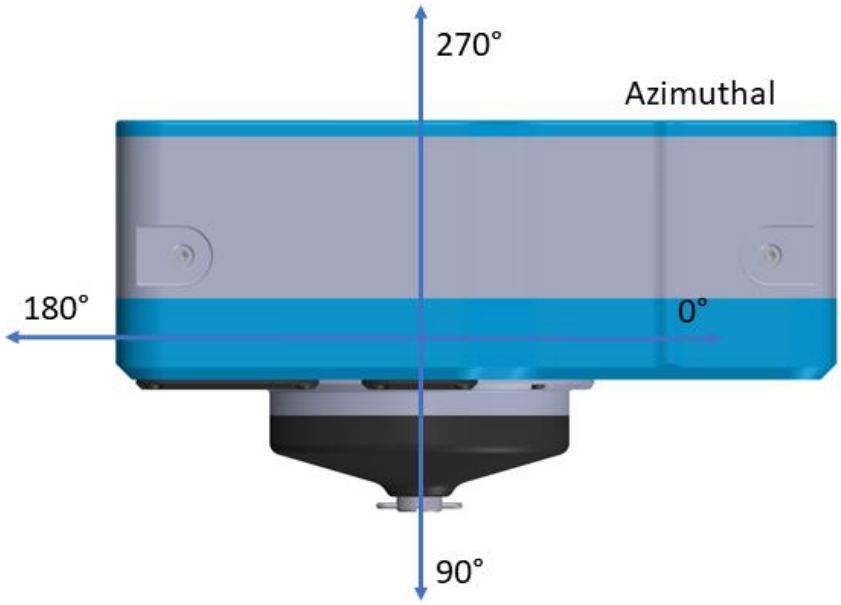


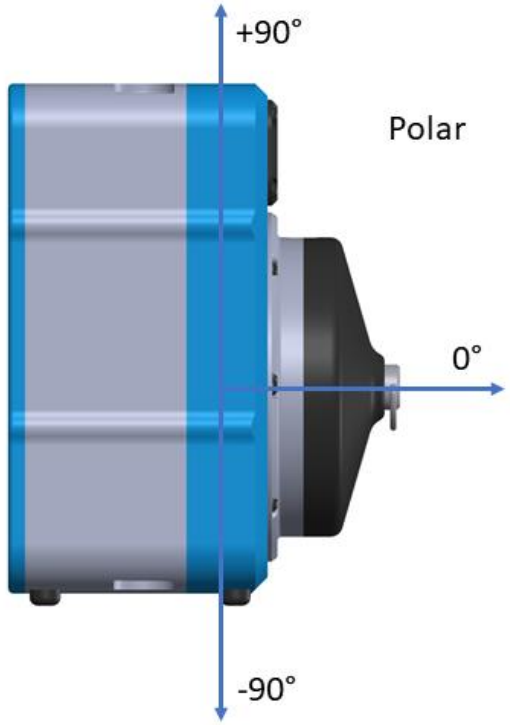


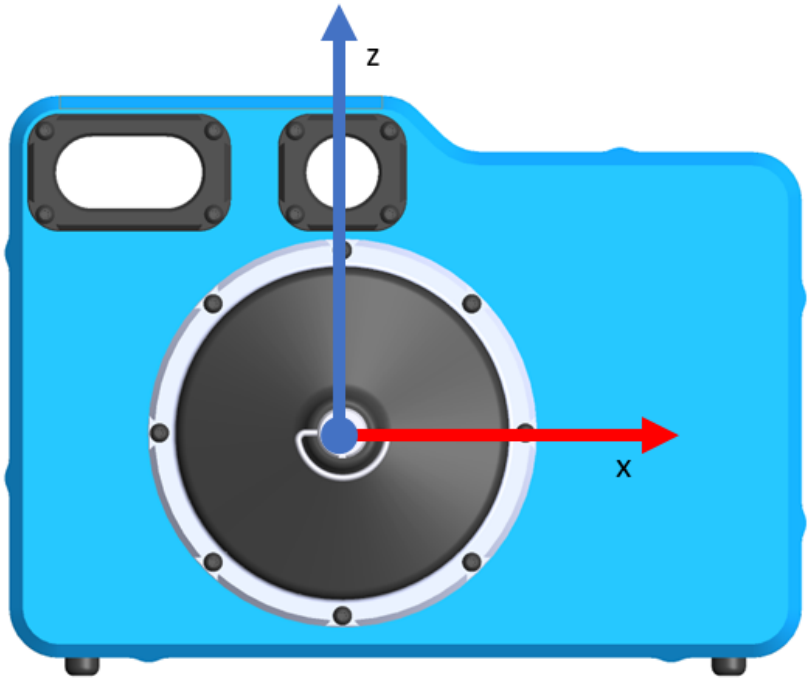

z
x

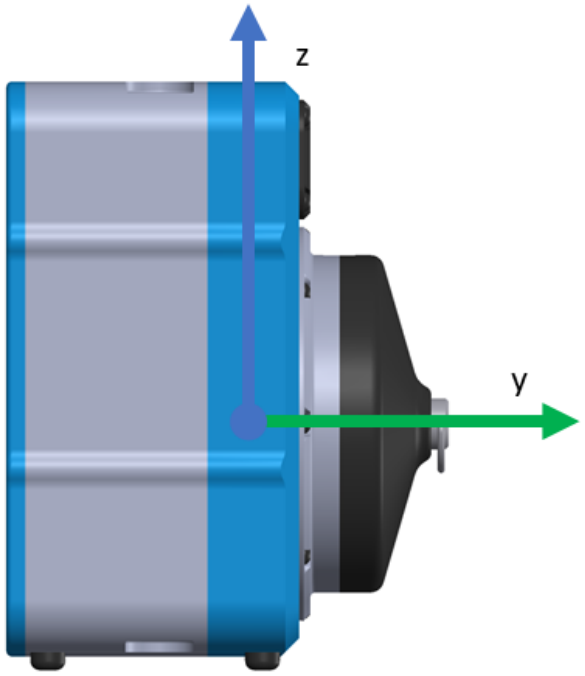

z
y

P100 Series

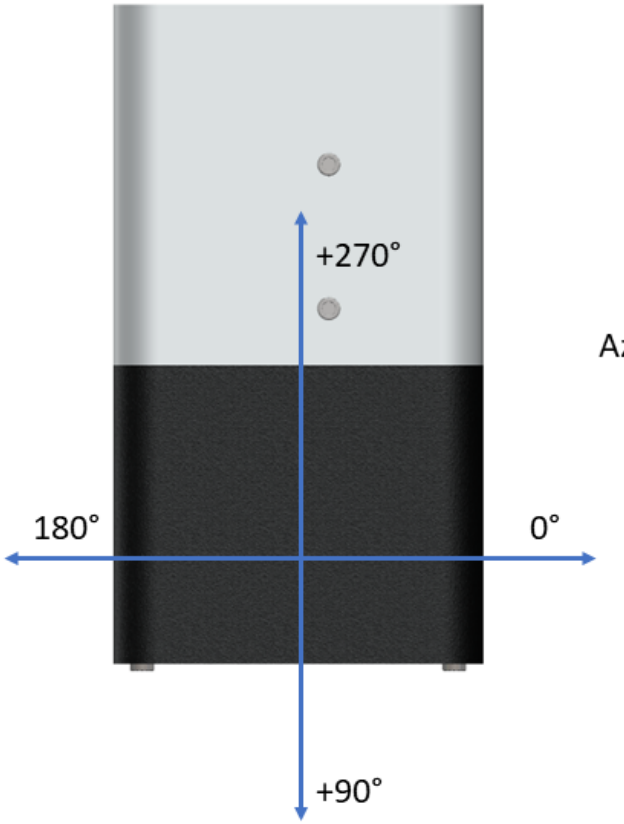




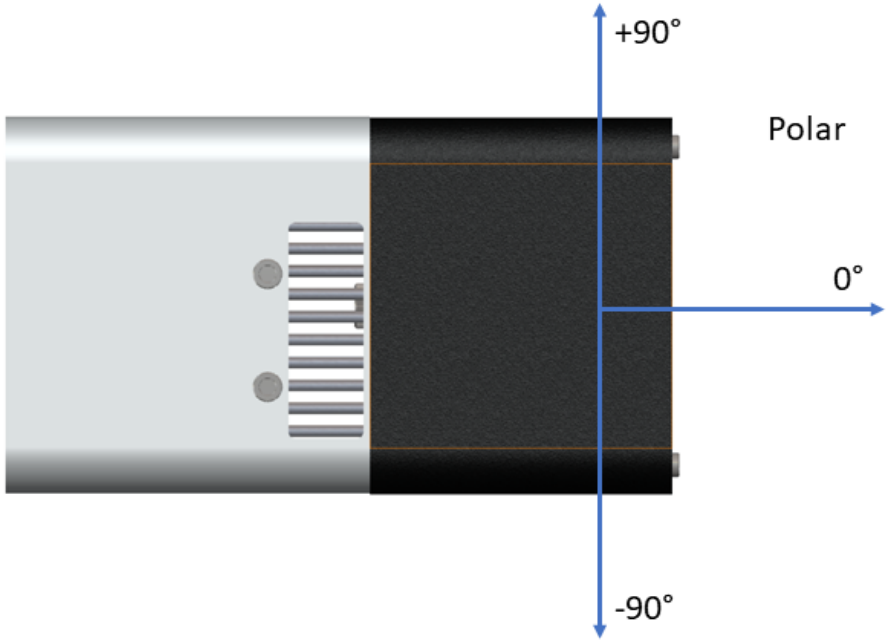

z
x

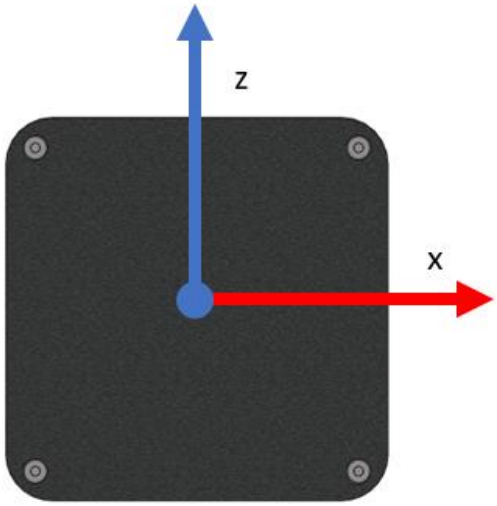

z
y

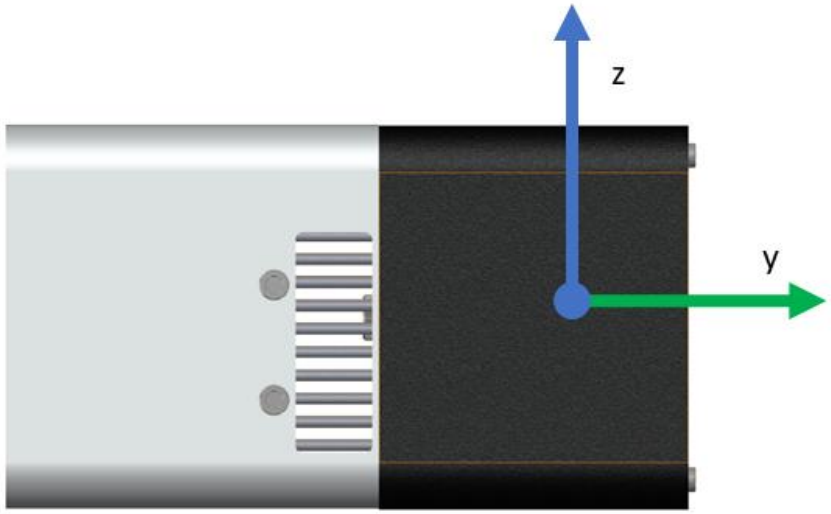
M100 / M400 Series



Azimuthal

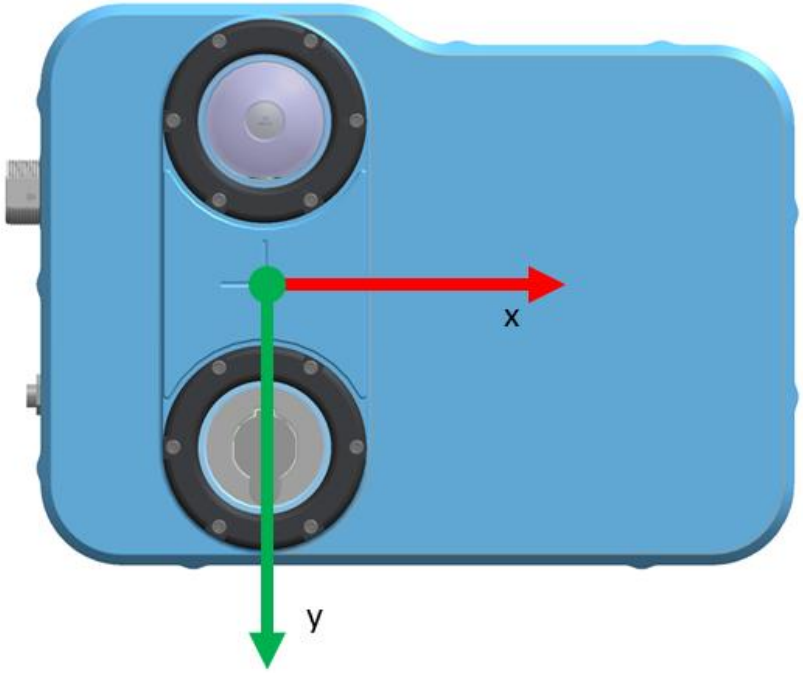


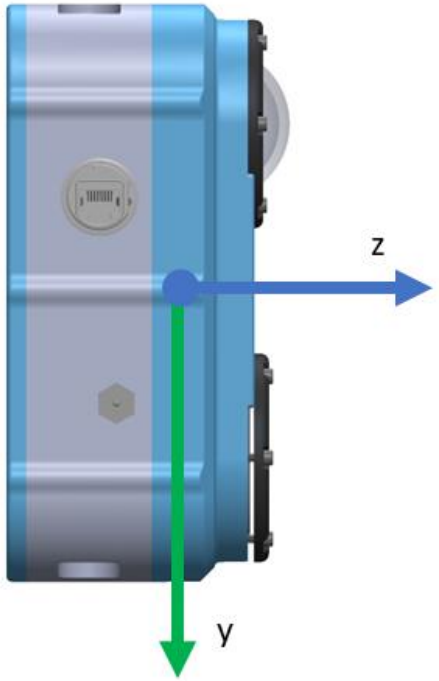

z
x


z
y

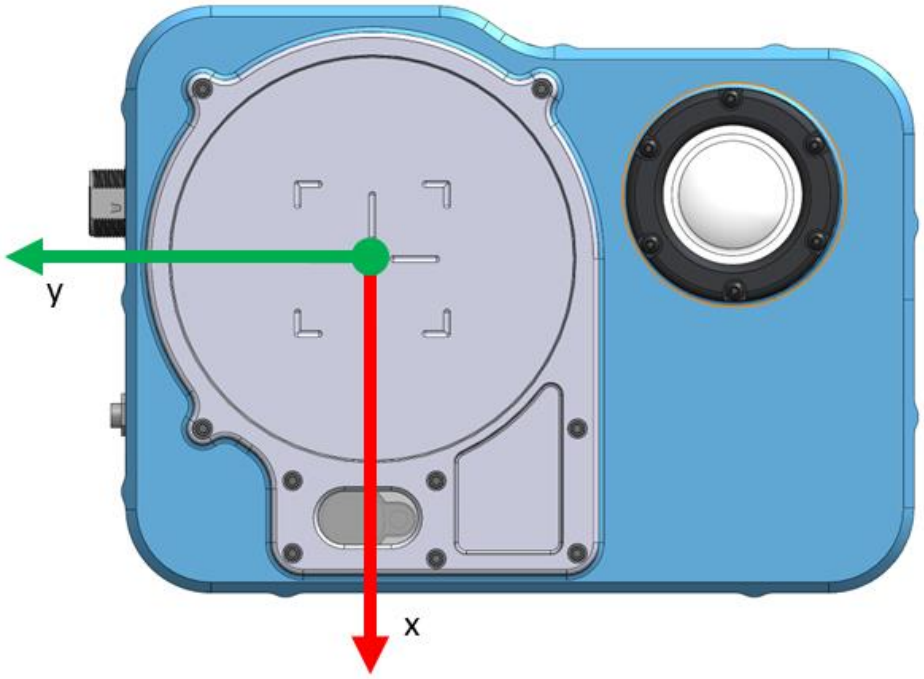
Appendix E – List-Mode Data Coordinate Systems

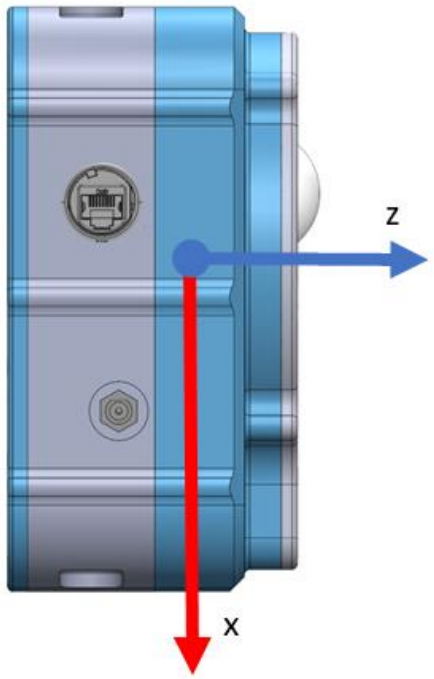
H100 Series



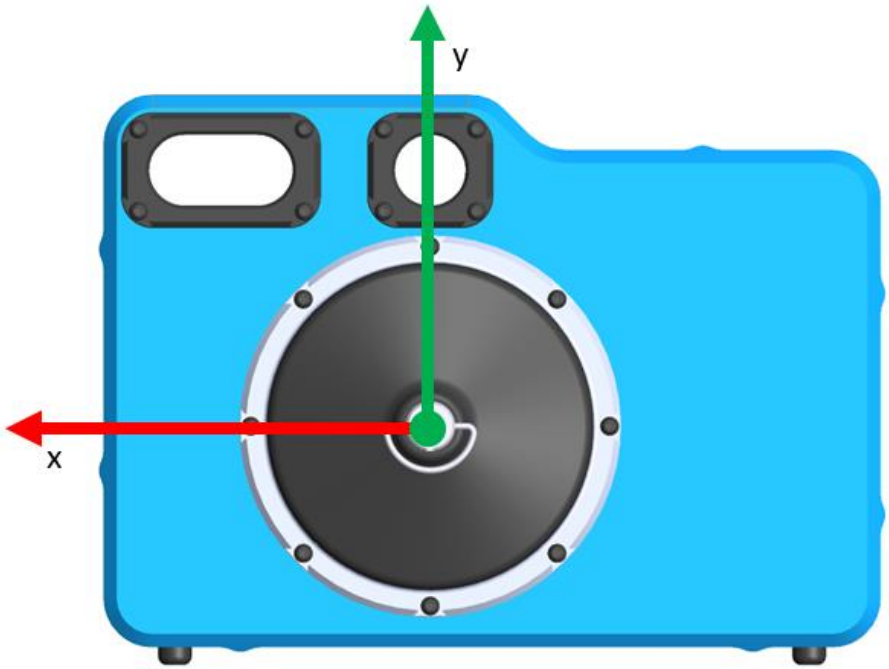


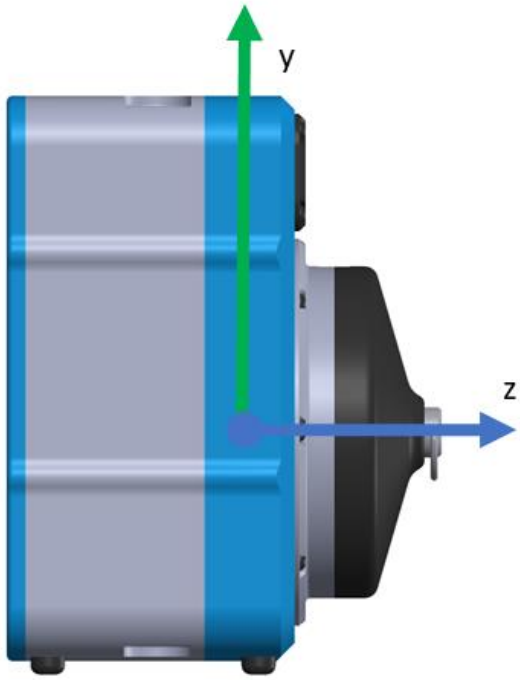
H400 Series



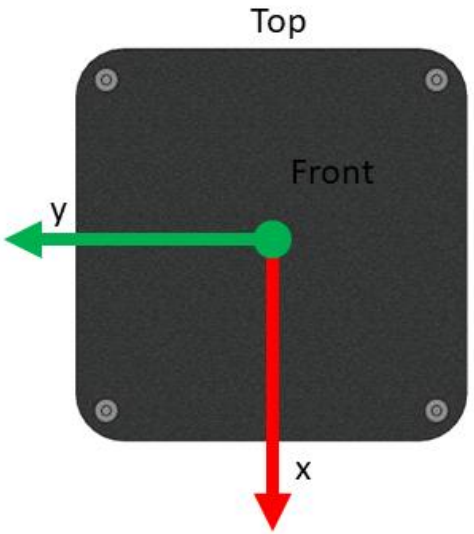


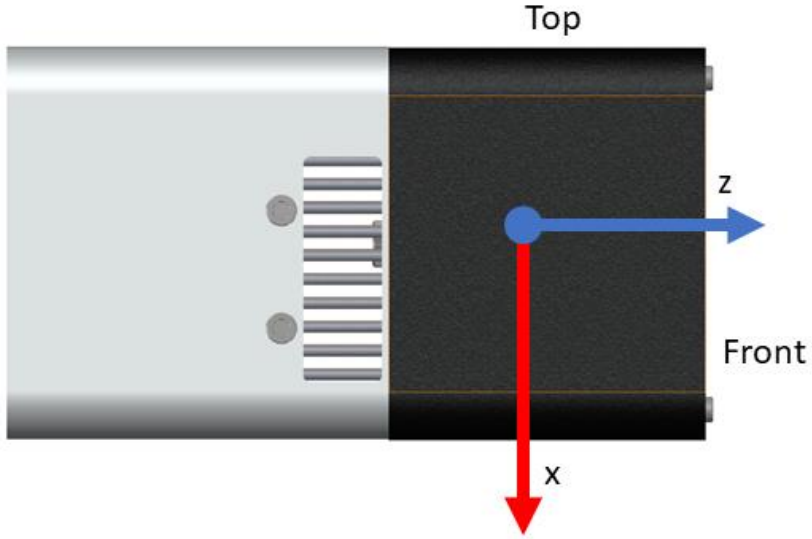
P100 Series





M100 / M400 Series





Appendix F – Status Information Definitions

Endpoint (/api/v1/status)	Type	Units	Description	Applicable Systems
config/CustomImageScale	boolean	N/A	Whether or not the system is using a custom image scale for rendering rad images	Imaging systems
config/Distance	float	meters	Source to detector distance currently set on the system	All systems
config/MaskPlugState	boolean	N/A	Whether or not the plug is installed in the P series system	P series systems
config/NearField	boolean	N/A	Whether or not the system is configured to account for near-field affects	Systems with optical camera
config/PresetTime	integer	seconds	Measurement time duration for fixed-time measurements (-1 if not enabled)	All systems
config/RepeatTime	integer	seconds	Measurement time duration for repeat measurements (-1 if not enabled)	All systems
config/UTCOffset	float	hours	Current UTC offset for local time	All systems
config/UsingLiveTime	boolean	N/A	Whether or not the system is using live time (instead of real time) for measurement timers	All systems
measurement/CountRate	integer	cps	Current count rate measured by the system	All systems
measurement/CurrentDirectory	string	N/A	Relative directory of currently recording measurement (empty string if not currently recording a measurement)	All systems
measurement/DeadTimeFraction	float	N/A	Fraction of time that the system is dead due to detector readout (range 0-1)	All systems
measurement/DoseRate	float	mRem/hr	Current dose rate being recorded by the system	All systems
measurement/ElapsedSeconds	string	seconds	Time elapsed in current measurement (measurement running) or elapsed time in most recently stopped measurement (measurement stopped) or time since search mode began. Will be live time if UsingLiveTime is true.	All systems

measurement/MeasurementMode	integer	N/A	0-stopped, 1-static measurement, 2-search mode	All systems
measurement/PreviousDirectory	string	N/A	Relative directory of previous recorded measurement (empty string if system has not yet finished recording a measurement)	All systems
measurement/TotalDose	float	mRem	Total dose recorded by detector in current measurement or the previously recorded measurement or total dose since entering search mode	All systems
network/RMSIPaddress	string	N/A	IP address the system uses to attempt connections to a GammAware server	RMS configured systems
network/Gateway	string	N/A	Default IPv4 gateway of the Ethernet port or USB-Ethernet adapter for systems that do not have a native Ethernet port	All systems
network/Ethernet	string	N/A	IPv4 IP address of the Ethernet port or USB-Ethernet adapter for systems that do not have a native Ethernet port	All systems
network/Netmask	string	N/A	IPv4 netmask of the Ethernet port or USB-Ethernet adapter for systems that do not have a native Ethernet port	All systems
state/BatteryLevel	string	%	Battery charge percentage - may report "Wrong DC input" if a power adapter with the incorrect output voltage is used	Systems with batteries
state/BiasStatus	integer	seconds	Time until system has biased up	All systems
state/CameraFault	boolean	N/A	Whether or not the camera is in a fault state	Systems with optical camera
state/DAQError	integer	N/A	Bitwise indicator that may describe several low-level failures	All systems
state/DiskFree	integer	MB	Amount of free space on the disk that on which the system is currently saving data	All systems
state/DiskFreeFraction	integer	%	Percentage of the disk that is free	All systems

state/DiskSpaceError	integer	N/A	0 - No error, 1 - Space low (List-mode data is no longer saved), 2 - Disk space critical (No new measurements or meta files saved)	All systems
state/MaskFault	boolean	N/A	Whether or not the CAI mask is in a fault state	Systems with spinning CAI mask
state/NearFieldOpticalTooClose	boolean	N/A	Whether or not the source distance is set close enough that is will significantly distort the optical image	Systems with optical camera
state/PowerSource	integer	N/A	0 - External power, 1 - internal battery	All systems
state/RelativeHumidity	integer	%	Percent relative humidity inside the detector system	All systems
state/StorageInternal	boolean	N/A	Whether or not the system is currently saving data to an internal drive	All systems
state/StorageMounted	boolean	N/A	Whether or not a data storage drive is currently accessible and in use by the system	All systems

Appendix G – Unit Information Definitions

Endpoint (/api/v1/unit)	Type	Units	Description	Applicable Systems
Config	string	N/A	Configuration identifier (Power,CBRN,Safeguards,EOD)	All systems
CustomSN	string	N/A	Customer supplied SN	All systems
Model	string	N/A	Model identifier for system	All systems
Name	string	N/A	Name of the system	All systems
SN	string	N/A	Serial number for the system	All systems
versions/Patch	string	N/A	Identifier for the latest software patch applied to system	All systems