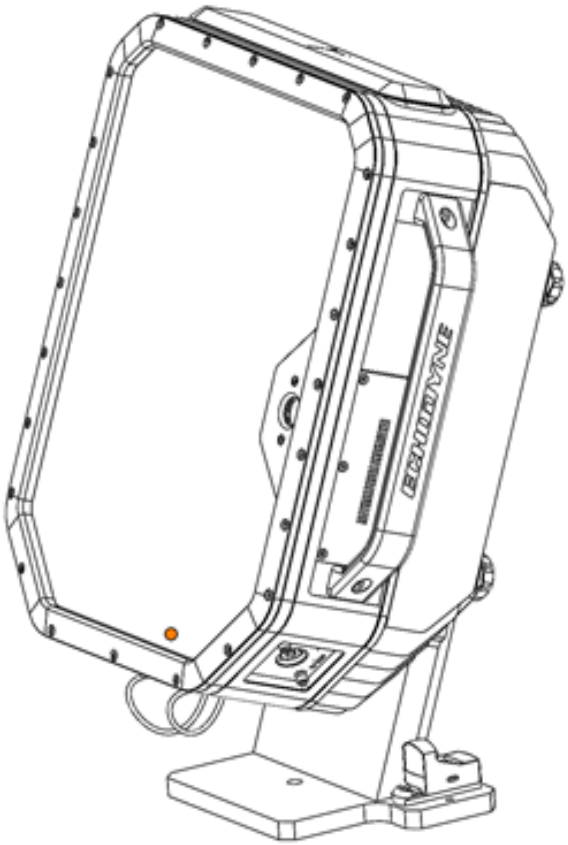
EchoShield Developer Manual

Radar API - Interface Definitions

EchoShield API v8.1.1 (For EchoShield SW Bundle v8.1)

Release Date: October 2025



ECHODYNE

Contents:

1 Introduction

1.1 API Version

This document details the EchoShield radar API version 8.1.1.

1.2 Purpose

This document comprises the Developer Manual for Echodyne's **EchoShield** radar platform.

Note: **EchoShield** is a product of Echodyne's **Jupiter** project. These names are interchangeable.

1.3 Scope

This document covers External, User-facing aspects of the EchoShield interface, referenced as RadarAPI, accessible by the following access roles:

- USER
- SUPER_USER
- FIELD_UPGRADE_USER

The following topical areas are covered in this manual:

- Ethernet - the physical interfaces' descriptions and behaviors
- Commands - request/response (REQ/RSP) exchanges from a User client to the Radar embedded software
- Parameters - the arguments to commands (requests)
- Data - the definitions associated with the interface's data products

1.4 Breaking Changes

With each release, elements of the EchoShield API may be changed in a ways that are potentially incompatible with client software. Changes of this nature are announced as early as possible, and will not take effect until at least the next major release of the EchoShield API.

Potentially breaking changes come in many forms, to name a few:

- API element removal
- Input parameter bounds tightening
- Output packet field bounds loosening

- Addition of a new required parameter in a command request

For each upcoming breaking change, a notice is post-fixed to the affected element's description in this document. The official list of API usage changes is found in a separate Release Notes document associated with this release of the EchoShield API.

1.5 Intended Audience

The audience for this document is software developers and advanced end users of the EchoShield radar platform.

2 Ethernet

2.1 1Gb Ethernet Interface

Ports, Services, and Protocols

The 1G Ethernet interface supports the standard Ethernet services of Ping (ICMP) and ARP.

The 1G Ethernet protocol requirements for the Radar supports Internet Protocol version 4 (IPv4).

The 1G Ethernet packet maximum transfer unit (MTU) size is set to 1500 bytes.

Upon initial delivery, the default IP address for this interface is defined in the parameters specification table: *1Gb Network Parameters Specification*.

Users can change these standard IP settings using the defined maintenance features through the 1G interface.

The Radar utilizes the following *Registered* Ports:

Registered Port Number	Service Name	Service Description
123	Network Time Protocol (NTP)	Protocol for clock synchronization between computer systems over a packet-switched network.
5514	Remote system event & status logging	UDP Syslog *server* receive port (RFC 5424).

The Radar-defined 1G *User* Ports are as follows:

User Port (unencrypted)	Service Name	Service Description
29978	Command	TCP Command & response service
29979	Status	TCP Status information stream
29981	Detections	TCP Detections data stream
29982	Tracks	TCP Track information stream
29984	Measurements	TCP Measurement information stream
29986	Beams	TCP Beam information stream
29988	Kinematics Output	TCP Kinematics output packet stream
29989	Kinematics Input	UDP Kinematics input stream (see *Streaming Kinematics*)
29977	Locator	UDP locator service

1Gb Network Parameters Specification

Note: All network changes require the radar to be in *Service mode*.

Parameter	Description	Default Value	Parameter Update Context
MAC Address	MAC Address of SoC Processor 1G Ethernet interface. Will be assigned an address in the range 34:4C:C8:00:75:30 - 34:4C:C8:00:EA:5F	34:4C:C8:xx:xx:xx	N/A
IP Address	IPv4 Address of SoC Processor 1G Ethernet interface	192.168.1.150	On change, restart interface
IP Subnet Mask	IPv4 Subnet Mask of SoC Processor 1G Ethernet interface	255.255.255.0	On change, restart interface
IP Default Gateway	IPv4 Gateway of SoC Processor 1G Ethernet interface	192.168.1.1	On change, restart interface
IP Network	IPv4 Network of SoC Processor 1G Ethernet interface	DERIVED FROM IP ADDRESS	NA
IP Primary Domain Name Server	IPv4 Primary DNS of SoC Processor 1G Ethernet interface	NOT SET	On change
IP Secondary Domain Name Server	IPv4 Secondary DNS of SoC Processor 1G Ethernet interface	NOT SET	On change
Hostname	IPv4 Hostname of SoC Processor 1G Ethernet interface	echoshield-01	On change
Search Domain	IPv4 Search Domain of SoC Processor 1G Ethernet interface	NOT SET	On change
NTP Primary Server IP Address	IPv4 NTP Primary Server IP Address of SoC Processor 1G Ethernet interface	NOT SET	On change
NTP Secondary Server IP Address	IPv4 NTP Secondary Server IP Address of SoC Processor 1G Ethernet interface	NOT SET	On change
NTP Tertiary Server IP Address	IPv4 NTP Tertiary Server IP Address of SoC Processor 1G Ethernet interface	NOT SET	On change
Syslog Host IP Address	IPv4 Address for Syslog Host	NO DEFAULT	On change, restart syslogd

continues on next page

Table 1 – continued from previous page

Parameter	Description	Default Value	Parameter Update Context
Locator	Radar Locator Port Number on 1G Ethernet Interface	29977	On change, restart radarlocator
Command	Radar Command Port Number on 1G Ethernet Interface	29978	On change, restart
Status	Radar Status Port Number on 1G Ethernet Interface	29979	On change, restart
Detections	Radar Detections Port Number on 1G Ethernet Interface	29981	On change, restart
Tracks	Radar Tracks Port Number on 1G Ethernet Interface	29982	On change, restart
Measurements	Radar Measurements Port Number on 1G Ethernet Interface	29984	On change, restart
Beams	Radar Beam Port Number on 1G Ethernet Interface	29986	On change, restart
Kinematics	Radar Kinematics Port Number on 1G Ethernet Interface	29988	On change, restart

2.2 10Gb Ethernet Interface

Ports, Services, and Protocols

The 10Gb interface provides Radar RDMap data to the user. The 10Gb interface is a hardware accelerated interface, only supporting UDP data over Internet Protocol version 4 (IPv4) network with no dynamic addressing, no security, and no network management support.

The 10Gb protocol layer is designed to support an MTU size of 9000 and jumbo frames of 9216 bytes.

Given the Radar data rates, using a standard 1500 MTU size leads to a very high packet per second rate limiting overall system performance. Ethernet switches, test devices, and sensor processing computers on the 10Gb network must support jumbo frames.

The 10Gb Ethernet interface supports the standard Ethernet services of Ping (ICMP) and ARP only. No support for Multicast (IGMP) or VLANs is provided. No support is provided for DHCP as the user's network for Radar RDMap data is assumed static. Domain Name Servers (DNS) and Domain Name Search Path are not supported.

10Gb IP settings are changed using Echodyne-defined control messages over the 1G Ethernet interface.

Upon initial delivery, the default IP address for this interface is defined by the control parameters listed on this page: *10Gb Network Parameters Specification*.

The Radar-defined 10Gb *User* Ports are as follows:

User Port Number	Service Name	Service Description
30980	RDMap	UDP range/Doppler maps (RDMap)

10Gb Network Parameters Specification

Parameter	Description	Default Value	Parameter Update Context
MAC Address	MAC Address of FPGA 10G Ethernet interface. Will be assigned an address in the range 34:4C:C8:00:75:30 - 34:4C:C8:00:EA:5F	34:4C:C8:xx:xx:xx	N/A
MTU Size	MTU (Packet) size of data on 10G Ethernet Interface	9000	Commit at next Reset / Power Cycle
IP Address	IPv4 Address of FPGA 10G Ethernet interface	172.16.10.20	On change, restart interface
IP Subnet Mask	IPv4 Subnet Mask of FPGA 10G Ethernet interface	255.255.255.0	On change, restart interface
IP Default Gateway	IPv4 Gateway of FPGA 10G Ethernet interface	172.16.10.1	On change, restart interface
RVmap Destination Port Number	10GbE Rvmap UDP Destination Port Number	30980	On change, restart interface
RVmap Destination Host IP Address	10GbE RVmap Destination Host IPv4 Address	172.16.10.10	On change, restart interface
Loopback Port Number	Loopback Port Number	30950	On change, restart interface

3 Commands - (REQ/RSP)

This section documents the Radar Commands that are exposed by the API for Users to configure and control the Radar.

3.1 Data Exchange Format

The data exchange format follows a modified version of JSON-RPC 2.0 where we drop the `jsonrpc` version indicator field.

A single command exchange consists of a User Request (REQ) followed by a Radar Response (RSP).

User Request messages are expected to be terminated by the character sequence: "\r\n" . Radar Responses will be terminated in the same way.

An example of a simple User REQ:

```
{
    "method": "method_name", // STRING: The name of the Radar method to invoke.
    "id": 1,                 // INTEGER: An incrementing integer message␣
↪identifier
}
```

Some commands contain parameters and will include the `params` object:

```
{
    "method": "method_name", // STRING: The name of the Radar method to invoke.
    "params": {},            // OBJECT: An object containing method parameters
    "id": 1,                 // INTEGER: An incrementing integer message␣
↪identifier
}
```

When the User Request succeeds, the Radar Response will be in a Success Response format, which is based on the "method" and "params" objects of the User Request.

Radar RSP for success example

```
{
    "result": {},  // OBJECT: The result of the invoked Radar method, possibly␣
↪empty.
    "id": 1,       // INTEGER: The same identifier corresponding to REQ
}
```

User Requests can fail for several reasons. In these cases, the Radar Response will be in the Error Response format, defined as follows:

```
{
    "error": {
        "code": 1,              // INTEGER: Error code
        "message": "REASON"     // STRING: Brief description of the error
    },
    "id": 1,                   // INTEGER: The same identifier corresponding to␣
↪REQ
}
```

The 'code' field of the error object indicates the type of error the radar encountered, while the 'message' field gives specific details about the error. Error codes are detailed in the table below:

Code	Category	Description
1	Fail	Generic failure code, see 'message' field for details.
-32700	ParseError	Missing or invalid 'method', 'id' or 'params' fields.
-32601	MethodNotFound	Command request JSON contains a 'method' that the radar does not recognize, or no 'method' at all.
-32602	InvalidParams	Invalid Parameters included in the 'params' field, such as values out of range, incorrect types, or missing required parameters.
-32603	InternalError	Failure to parse incoming command due to invalid JSON or lack of system resources.
-32000	AlreadyConnected	Sent to all client connections made beyond the first before the connection is closed.

3.2 Role Authentication

Each command described in the API lists the **Roles** for which the command can be used.

Any connected client is automatically authenticated to the USER role. The USER role includes the *get_idn* command which allows the client to get identifying information about the connected radar as well as the *authorize_role* and *get_session_id* commands which allow authorization of additonal roles.

Authenticating a non-USER Role requires two pieces of information:

- A public key file provided by Echodyne for a specific role
- A Radar Command session ID

Session IDs are generated once per network connection to the Command port on the Radar. If the client disconnects from the radar, the authorized roles for that session are no longer authorized, and must be reauthorized with a new session ID.

To authenticate, client apps should

- Query for the current session ID using the *get_session_id* command.

- Encrypt the session ID using the role-specific public key file contents and the RSAES-PKCS1-v1_5 encryption scheme.
- Encode the encrypted bytes as Base64 ASCII.
- Include the resulting characters in the `key` field of the *authorize_role* command.

Attempting to use a command that is not allowed under the currently authenticated roles will cause the *AUTH* error to be returned. See the documenation for the command to learn which role needs to be authenticated to use that command. If there is more than one role for a command, the command is authenticated if at least one of the listed roles is authenticated.

Role authorization Payload example

```
{
     "method": "authorize_role",
     "params": {
         // STRING: The role to authorize
         "role": "SUPER_USER",
         // STRING: The UUID for this session
         "uuid": "47578539-4850-459a-8b81-3c249a82132d",
         // STRING: The Base64-encoded bytes represeting the encrypted UUID
         "key": "DtBkyywedFaUo13OcjnXFXQv+72l1YlakHP9e...TRUNCATED=="
     },
     "id": 1
}
```

Python reference for deriving authorization key

```
def get_auth_payload(key_path: str, session_id: str) -> str:
    """Get authentication string based on session_id"""
    import base64
    from cryptography.hazmat.primitives.serialization import load_pem_public_key
    from cryptography.hazmat.primitives.asymmetric.padding import PKCS1v15

    with open(key_path, "rb") as public_key:
        public_key_bytes = public_key.read()
        public_key = load_pem_public_key(public_key_bytes)
        encrypted_bytes = public_key.encrypt(session_id.encode(), PKCS1v15())
        b64_bytes = base64.standard_b64encode(encrypted_bytes)
        b64_string = b64_bytes.decode()
    return b64_string
```

User Roles

Authorizing a role grants access to the set of commands and parameters associated with that role. All authorizable roles and a high-level description of what API is granted access per role are listed in the table below:

Role	API Made Accessible
USER (authorized by default)	All API neccessary to set up and run the radar.
SUPER_USER	Configuration commands/parameters for fine-tuning of radar operation.
FIELD_UPGRADE_USER	All commands related to field-upgrades of the radar firmware.

3.3 Missions

The RadarApp command API provides commands that allow for the listing (*list_missions*), licensing (*license_mission*), and selection (*configure_mission*) of missions. Selection of a mission configures the radar in a fashion that optimizes its ability to detect, track, and classify a particular class of targets, this includes setting the default values of various user-facing RadarApp parameters to mission-specific values.

This section describes each of the configurable missions.

C-UAS_1

The C-UAS_1 mission is optimized to provide air space situational awareness against group 1 UAS at ranges less than 4km. The mission has been designed to provide a minimum track acquisition range of 150 meters, a minimum blind range on tracked targets of 100 meters, and a 50% track acquisition range on a -20 dBsm target of 2.5km. This mission is expected to be able to track ground targets in a limited capacity. The C-UAS_1 mission is recommended for customers looking to track group 1 UAS and smaller targets.

C-UAS_2

The C-UAS_2 mission is optimized to provide air space situational awareness against group 1-3 UAS at ranges up to 15km. The mission has been designed to provide a minimum track acquisition range of 500 meters, a minimum blind range on tracked targets of 100 meters, and a 50% track acquisition range on a -20 dBsm target of 2.5km. This mission is expected to be able to track ground targets in a limited capacity. The C-UAS_2 mission is recommended for customers looking to track group 1-3 UAS or targets of similar size and speed.

DISMOUNT

The DISMOUNT mission is optimized to provide ground situational awareness at ranges up to 15 km. The mission has been designed to provide a 50% track acquisition range on a -3 dBsm target (human) of 8km. This mission is not expected to be able to detect and track airborne targets. The DISMOUNT mission is recommended for customers interested in tracking targets on the ground.

COASTAL_C-UAS

The COASTAL_C-UAS mission is optimized to provide air space situational awareness against group 1-3 UAS operating over water at ranges up to 15km. The mission set uses proprietary algorithms to estimate the direction and velocity of waves and reduce the number of detections, and ultimately tracks, generated by those waves. The mission has been designed to provide a minimum track acquisition range of 500 meters, a minimum blind range on tracked targets of 100 meters, and a 50% track acquisition range on a -20 dBsm target of 2.5km. The COASTAL_C-UAS mission is recommended for customers looking to track group 1-3 UAS or targets of similar size and speed that are operating over water.

3.4 System Commands

General system commands and querying

- *authorize_role*
- *clear_log_overrides*
- *delete_existing_logs*
- *disable_log_category*
- *enable_log_category*
- *enter_service*
- *exit_fault_mode*
- *exit_service*
- *get_idn*
- *get_log_level*
- *get_net*
- *get_net_deltas*
- *get_radar_logs*
- *get_radar_mode*
- *get_session_id*
- *get_sys*
- *get_sys_deltas*
- *get_systime*
- *get_time_info*
- *get_versions*
- *list_all_log_categories*
- *list_authorized_roles*
- *list_deprecation_roadmap*
- *list_enabled_log_categories*
- *query_fault*
- *reboot*
- *set_log_level*
- *set_net*

- *set_sys*
- *set_systime*

authorize_role

Authorize a role

This command is available with authorization of the following access role(s):

- USER (default)

User Request Parameters

Name	Description	Type	Constraints
role	Role to authorize	enum	*ROLE_LEVELS*
uuid	session UUID	string	ASCII characters
key	Encrypted body	string	ASCII characters

Example Request

RadarIO Prompt Syntax

```
authorize_role role ECHODYNE_ONLY uuid <session UUID> key <Encrypted body>
```

RadarIO Script Syntax

```
client.authorize_role(role=ECHODYNE_ONLY, uuid=<session UUID>, key=<Encrypted␣
↪body>)
```

JSON Syntax (as written to control port)

```
{
  "method": "authorize_role",
  "id": 105,
  "params": {
    "role": "ECHODYNE_ONLY", "uuid": "<session UUID>", "key": "<Encrypted body>"
  }
}
```

Example Response

```
{"id": 105, "result": {}}
```

clear_log_overrides

Clear the effects of *enable_log_category*, *disable_log_category*, and *set_log_level*.

This command is available with authorization of the following access role(s):

- SUPER_USER

Example Request

RadarIO Prompt Syntax

```
clear_log_overrides
```

RadarIO Script Syntax

```
client.clear_log_overrides()
```

JSON Syntax (as written to control port)

```
{"method": "clear_log_overrides", "id": 100}
```

Example Response

```
{"id": 100, "result": {}}
```

delete_existing_logs

Delete all existing system logs from disk.

This command is available with authorization of the following access role(s):

- SUPER_USER

Example Request

RadarIO Prompt Syntax

```
delete_existing_logs
```

RadarIO Script Syntax

```
client.delete_existing_logs()
```

JSON Syntax (as written to control port)

```
{"method": "delete_existing_logs", "id": 134}
```

Example Response

```
{"id": 134, "result": {}}
```

disable_log_category

Disable radar syslog category. NOTE: changes to enabled log categories made with this command will be in effect until the *clear_log_overrides* command is issued, or the radar is rebooted.

This command is available with authorization of the following access role(s):

- SUPER_USER

User Request Parameters

Name	Description	Type	Constraints
category	The case-insensitive name of the log category to disable	enum	*LOG_CATEGORIES*

Example Request

RadarIO Prompt Syntax

```
disable_log_category category UNIT_TEST
```

RadarIO Script Syntax

```
client.disable_log_category(category=UNIT_TEST)
```

JSON Syntax (as written to control port)

```
{
  "method": "disable_log_category",
  "id": 99,
  "params": {"category": "UNIT_TEST"}
}
```

Example Response

```
{"id": 99, "result": {}}
```

enable_log_category

Enable radar syslog category. NOTE: changes to enabled log categories made with this command will be in effect until the *clear_log_overrides* command is issued, or the radar is rebooted.

This command is available with authorization of the following access role(s):

- SUPER_USER

User Request Parameters

Name	Description	Type	Constraints
category	The case-insensitive name of the log category to enable	enum	*LOG_CATEGORIES*

Example Request

RadarIO Prompt Syntax

```
enable_log_category category TX1_CTRL
```

RadarIO Script Syntax

```
client.enable_log_category(category=TX1_CTRL)
```

JSON Syntax (as written to control port)

```
{
  "method": "enable_log_category", "id": 98, "params": {"category": "TX1_CTRL"}
}
```

Example Response

```
{"id": 98, "result": {}}
```

enter_service

Enter Service mode. The radar must be in service mode to update any parameters with a 'Modifiable Mode' of 'Service' using the *set*, *set_net*, and *set_sys* commands.

This command is available with authorization of the following access role(s):

- USER (default)

Example Request

RadarIO Prompt Syntax

```
enter_service
```

RadarIO Script Syntax

```
client.enter_service()
```

JSON Syntax (as written to control port)

```
{"method": "enter_service", "id": 117}
```

Example Response

```
{"id": 117, "result": {}}
```

exit_fault_mode

Exits clearable fault mode

This command is available with authorization of the following access role(s):

- USER (default)

Example Request

RadarIO Prompt Syntax

```
exit_fault_mode
```

RadarIO Script Syntax

```
client.exit_fault_mode()
```

JSON Syntax (as written to control port)

```
{"method": "exit_fault_mode", "id": 121}
```

Example Response

```
{"id": 121, "result": {}}
```

exit_service

Exit Service mode and return to Operation mode, applying changes made to service-modifiable parameters.

This command is available with authorization of the following access role(s):

- USER (default)

Example Request

RadarIO Prompt Syntax

```
exit_service
```

RadarIO Script Syntax

```
client.exit_service()
```

JSON Syntax (as written to control port)

```
{"method": "exit_service", "id": 118}
```

Example Response

```
{"id": 118, "result": {}}
```

get_idn

Get basic identifying information, such as serial numbers and API versions.

This command is available with authorization of the following access role(s):

- USER (default)

Radar Response Parameters

Name	Description	Type	Constraints
org	Company name	string	"Echodyne"
product	Product name	string	"EchoShield"
serial	Serial number	string	ASCII characters
bundle_version	Upgrade package version	string	<Semantic Version>-<Build Number>
api_version	Radar API version	string	Semantic Version
MAC_1G	1G MAC address	string	MAC address
MAC_10G	10G MAC address	string	MAC address
gnss_module_serial	GNSS module serial number	string	ASCII characters
enhanced_accelerometer_cal	True if this unit has undergone enhanced accelerometer calibration.	boolean	[True, False]

Example Request

RadarIO Prompt Syntax

```
get_idn
```

RadarIO Script Syntax

```
client.get_idn()
```

JSON Syntax (as written to control port)

```
{"method": "get_idn", "id": 91}
```

Example Response


```
{
  "id": 91,
  "result": {
    "org": "<Company name>",
    "product": "<Product name>",
    "serial": "<Serial number>",
    "bundle_version": "<Upgrade package version>",
    "api_version": "<Radar API version>",
    "MAC_1G": "<1G MAC address>",
    "MAC_10G": "<10G MAC address>",
    "gnss_module_serial": "<GNSS module serial number>",
    "enhanced_accelerometer_cal": "<True if this unit has undergone enhanced␣
↪accelerometer calibration.>"
  }
}
```

get_log_level

Get the current radar syslog level.

This command is available with authorization of the following access role(s):

- SUPER_USER

Radar Response Parameters

Name	Description	Type	Constraints
level	Active logging level	enum	*LOG_LEVELS*

Example Request

RadarIO Prompt Syntax

```
get_log_level
```

RadarIO Script Syntax

```
client.get_log_level()
```

JSON Syntax (as written to control port)

```
{"method": "get_log_level", "id": 96}
```

Example Response

```
{"id": 96, "result": {"level": "<Active logging level>"}}
```

get_net

Get network configuration parameters

This command is available with authorization of the following access role(s):

- USER (default)

User Request Parameters

Name	Description	Type	Constraints
retrieve	**Optional** Parameters to retrieve	string[]	ASCII characters Maximum 256 elements

Radar Response Parameters

Dictionary of specified radar parameter values.

Example Request

RadarIO Prompt Syntax

```
get_net retrieve [address_1G, command_port]
```

RadarIO Script Syntax

```
client.get_net(retrieve=['address_1G', 'command_port'])
```

JSON Syntax (as written to control port)

```
{
  "method": "get_net",
  "id": 111,
  "params": {"retrieve": ["address_1G", "command_port"]}
}
```

Example Response


```
{"id": 111, "result": {"address_1G": "192.168.1.151", "command_port": 29978}}
```

get_net_deltas

Get a list of all network configuration parameters that are not set to their default values.

This command is available with authorization of the following access role(s):

- USER (default)

Radar Response Parameters

Dictionary of parameters that are not set at their default values.

Example Request

RadarIO Prompt Syntax

```
get_net_deltas
```

RadarIO Script Syntax

```
client.get_net_deltas()
```

JSON Syntax (as written to control port)

```
{"method": "get_net_deltas", "id": 112}
```

Example Response

```
{
  "id": 112,
  "result": {"address_1G": "192.168.1.151", "netmask_1G": "255.255.254.0"}
}
```

get_radar_logs

Get a base64-encoded string of a gzipped-tarball containing the radar's system logs.

This command is available with authorization of the following access role(s):

- USER (default)

Radar Response Parameters

Name	Description	Type	Constraints
logs_data	Base64 encoded data of gzipped-tarball of radar system logs.	string	ASCII characters

Example Request

RadarIO Prompt Syntax

```
get_radar_logs
```

RadarIO Script Syntax

```
client.get_radar_logs()
```

JSON Syntax (as written to control port)

```
{"method": "get_radar_logs", "id": 133}
```

Example Response

```
{
  "id": 133,
  "result": {
    "logs_data": "<Base64 encoded data of gzipped-tarball of radar system logs.>"
  }
}
```

get_radar_mode

Get current radar mode

This command is available with authorization of the following access role(s):

- USER (default)

Radar Response Parameters

Name	Description	Type	Constraints
mode	Current mode	enum	*RADAR_MODES*

Example Request

RadarIO Prompt Syntax

```
get_radar_mode
```

RadarIO Script Syntax

```
client.get_radar_mode()
```

JSON Syntax (as written to control port)

```
{"method": "get_radar_mode", "id": 119}
```

Example Response

```
{"id": 119, "result": {"mode": "<Current mode>"}}
```

get_session_id

Get session id of the current connected session (used for role authorization).

This command is available with authorization of the following access role(s):

- USER (default)

Radar Response Parameters

Name	Description	Type	Constraints
session_id	Unique command session ID	string	ASCII characters

Example Request

RadarIO Prompt Syntax

```
get_session_id
```

RadarIO Script Syntax

```
client.get_session_id()
```

JSON Syntax (as written to control port)

```
{"method": "get_session_id", "id": 104}
```

Example Response

```
{"id": 104, "result": {"session_id": "<Unique command session ID>"}}
```

get_sys

Get system configuration parameters

This command is available with authorization of the following access role(s):

- USER (default)

User Request Parameters

Name	Description	Type	Constraints
retrieve	**Optional** Parameters to retrieve	string[]	ASCII characters Maximum 256 elements

Radar Response Parameters

Dictionary of specified radar parameter values.

Example Request

RadarIO Prompt Syntax

```
get_sys retrieve [loglevel]
```

RadarIO Script Syntax

```
client.get_sys(retrieve=['loglevel'])
```

JSON Syntax (as written to control port)

```
{"method": "get_sys", "id": 108, "params": {"retrieve": ["loglevel"]}}
```

Example Response

```
{"id": 108, "result": {"loglevel": "debug"}}
```

get_sys_deltas

Get a list of all system configuration parameters that are not set to their default values.

This command is available with authorization of the following access role(s):

- USER (default)

Radar Response Parameters

Dictionary of parameters that are not set at their default values.

Example Request

RadarIO Prompt Syntax

```
get_sys_deltas
```

RadarIO Script Syntax

```
client.get_sys_deltas()
```

JSON Syntax (as written to control port)

```
{"method": "get_sys_deltas", "id": 109}
```

Example Response

```
{"id": 109, "result": {"loglevel": "debug"}}
```

get_systime

Get the radar's operating system time and time source information.

This command is available with authorization of the following access role(s):

- USER (default)

Radar Response Parameters

Name	Description	Type	Constraints
time_unix_ns	System time in nanoseconds from the Unix epoch	integer	'int64_t' range
time_reference	Active time source reported by chronyd. Time sources are prioritized as follows: NMEA+PPS -> NTP[1-3]. Note that PPS and NMEA are both sourced from the internal GPS and work in tandem to form an accurate time source.	enum	• NTP1 • NTP2 • NTP3 • RESERVED • NMEA • PPS • NONE
sources	Dictionary of time sources detected by chronyd	*systime_sou*	

systime_sources

Name	Description	Type	Constraints
PPS	PPS (from internal GPS) is active and valid, see *gps_time_disable*	boolean	[True, False]
NMEA	NMEA (from internal GPS) is active and valid. Note that when NMEA+PPS is used as a time source, chronyd reports that PPS is active and NMEA is unreachable, in reality both sources are active.	boolean	[True, False]
NTP1	NTP1 is active and valid, see *ntp_primary_1G*	boolean	[True, False]

continues on next page

Table 14 – continued from previous page

Name	Description	Type	Constraints
NTP2	NTP2 is active and valid, see *ntp_secondary_1G*	boolean	[True, False]
NTP3	NTP3 is active and valid, see *ntp_tertiary_1G*	boolean	[True, False]

Example Request

RadarIO Prompt Syntax

```
get_systime
```

RadarIO Script Syntax

```
client.get_systime()
```

JSON Syntax (as written to control port)

```
{"method": "get_systime", "id": 102}
```

Example Response

```
{
  "id": 102,
  "result": {
    "time_unix_ns": "<System time in nanoseconds from the Unix epoch>",
    "time_reference": "<Active time source reported by chronyd. Time sources are␣
↪prioritized as follows: NMEA+PPS -> NTP[1-3]. Note that PPS and NMEA are both␣
↪sourced from the internal GPS and work in tandem to form an accurate time␣
↪source.>",
    "sources": "<Dictionary of time sources detected by chronyd>"
  }
}
```

get_time_info

Get detailed timekeeping info from the chrony daemon. Note that the system time reported by chronyd will only be valid if the radar has an active connection to a valid NTP / GPS (NMEA+PPS) time source. The *get_systime* command can be used to retrieve the radar's actual system time.

This command is available with authorization of the following access role(s):

- SUPER_USER

Radar Response Parameters

Name	Description	Type	Constraints
tracking	Result of *chronyc tracking* command	string	ASCII characters
sources	Result of *chronyc sources* command	string	ASCII characters
sourcestats	Result of *chronyc sourcestats* command	string	ASCII characters

Example Request

RadarIO Prompt Syntax

```
get_time_info
```

RadarIO Script Syntax

```
client.get_time_info()
```

JSON Syntax (as written to control port)

```
{"method": "get_time_info", "id": 103}
```

Example Response

```
{
  "id": 103,
  "result": {
    "tracking": "<Result of `chronyc tracking` command>",
    "sources": "<Result of `chronyc sources` command>",
```

(continues on next page)

(continued from previous page)

```
    "sourcestats": "<Result of `chronyc sourcestats` command>"
  }
}
```

`get_versions`

Get engineering version information, such as software component build strings and hardware revision numbers.

This command is available with authorization of the following access role(s):

- SUPER_USER

Radar Response Parameters

Name	Description	Type	Constraints
fpga_build	FPGA build version	string	Build string
platform	Hardware platform that hps-fw is running on	enum	*PLATFORMS*
bundle_build	Upgrade package build string	string	Build string
bsp_build	BSP build version	string	Build string
hps_build	HPS build version	string	Build string
mapcom_serial	MAPCOM serial number	string	ASCII characters
mapcom_revision	MAPCOM revision	string	ASCII characters
exciter_serial	Exciter serial number	string	ASCII characters
exciter_revision	MAPCOM revision	string	ASCII characters
tx0_serial	Tx0 serial number	string	ASCII characters
tx0_revision	MAPCOM revision	string	ASCII characters
tx1_serial	Tx1 serial number	string	ASCII characters
tx1_revision	MAPCOM revision	string	ASCII characters
antenna_serial	Antenna serial number	string	ASCII characters
jaxa_version	JAXA map version	string	ASCII characters
radar_app_sim_versio n	Compiled RadarAppSim version	string	ASCII characters
MAC_40G	40G MAC address	string	MAC Address
fused	Fused setting	boolean	[True, False]
production_mode	Production mode setting	boolean	[True, False]
hw_revision	Internal use	string	[Block0, Block1]
som_uuid	System on Module Universally Unique ID	string	ASCII characters
ocxo_serial	OCXO serial number	string	ASCII characters
ocxo_revision	OCXO revision	string	ASCII characters
ocxo_type	OCXO type	string	ASCII characters
multiclass_classifie r_version	Multiclass classifier version	string	ASCII characters

Example Request

RadarIO Prompt Syntax

```
get_versions
```

RadarIO Script Syntax

```
client.get_versions()
```

JSON Syntax (as written to control port)

```
{"method": "get_versions", "id": 92}
```

Example Response

```
{
  "id": 92,
  "result": {
    "fpga_build": "<FPGA build version>",
    "platform": "<Hardware platform that hps-fw is running on>",
    "bundle_build": "<Upgrade package build string>",
    "bsp_build": "<BSP build version>",
    "hps_build": "<HPS build version>",
    "mapcom_serial": "<MAPCOM serial number>",
    "mapcom_revision": "<MAPCOM revision>",
    "exciter_serial": "<Exciter serial number>",
    "exciter_revision": "<MAPCOM revision>",
    "tx0_serial": "<Tx0 serial number>",
    "tx0_revision": "<MAPCOM revision>",
    "tx1_serial": "<Tx1 serial number>",
    "tx1_revision": "<MAPCOM revision>",
    "antenna_serial": "<Antenna serial number>",
    "jaxa_version": "<JAXA map version>",
    "radar_app_sim_version": "<Compiled RadarAppSim version>",
    "MAC_40G": "<40G MAC address>",
    "fused": "<Fused setting>",
    "production_mode": "<Production mode setting>",
    "hw_revision": "<Internal use>",
    "som_uuid": "<System on Module Universally Unique ID>",
    "ocxo_serial": "<OCXO serial number>",
    "ocxo_revision": "<OCXO revision>",
    "ocxo_type": "<OCXO type>",
```

(continues on next page)

(continued from previous page)

```
    "multiclass_classifier_version": "<Multiclass classifier version>"
  }
}
```

list_all_log_categories

List all radar syslog categories.

This command is available with authorization of the following access role(s):

- SUPER_USER

Radar Response Parameters

Name	Description	Type	Constraints
log_categories	Array of all tracked log categories	enum[]	*LOG_CATEGORIES*

Example Request

RadarIO Prompt Syntax

```
list_all_log_categories
```

RadarIO Script Syntax

```
client.list_all_log_categories()
```

JSON Syntax (as written to control port)

```
{"method": "list_all_log_categories", "id": 95}
```

Example Response

```
{
  "id": 95,
  "result": {"log_categories": "<Array of all tracked log categories>"}
}
```

`list_authorized_roles`

List all access-control roles authorized in the current command session.

This command is available with authorization of the following access role(s):

- USER (default)

Radar Response Parameters

Name	Description	Type	Constraints
roles	Array of currently authorized roles	enum[]	*ROLE_LEVELS*

Example Request

RadarIO Prompt Syntax

```
list_authorized_roles
```

RadarIO Script Syntax

```
client.list_authorized_roles()
```

JSON Syntax (as written to control port)

```
{"method": "list_authorized_roles", "id": 94}
```

Example Response

```
{"id": 94, "result": {"roles": "<Array of currently authorized roles>"}}
```

`list_deprecation_roadmap`

Request deprecation roadmap entries.

This command is available with authorization of the following access role(s):

- USER (default)

User Request Parameters

Name	Description	Type	Constraints
index	**Optional** Index of deprecation	integer	'uint16_t' range
count	**Optional** Number to receive	integer	$x \leq 50$

Radar Response Parameters

Name	Description	Type	Constraints
deprecations		*deprecations*	

`deprecations`

Name	Description	Type	Constraints
deprecation_warning _array	Array of deprecation warnings	*deprecation_warn* []	Maximum 50 elements
total	Number of total deprecation warnings	integer	'uint16_t' range

deprecation_warning

Name	Description	Type	Constraints
deprecation	type of deprecation	string	ASCII characters
name	name of deprecation for reference	string	ASCII characters
announced_version	version of announcement	string	ASCII characters
effective_version	version of effectiveness	string	ASCII characters
description	short description	string	ASCII characters

Example Request

RadarIO Prompt Syntax

```
list_deprecation_roadmap index 16833 count 45
```

RadarIO Script Syntax

```
client.list_deprecation_roadmap(index=16833, count=45)
```

JSON Syntax (as written to control port)

```
{
  "method": "list_deprecation_roadmap",
  "id": 132,
  "params": {"index": 16833, "count": 45}
}
```

Example Response

```
{"id": 132, "result": {"deprecations": "<deprecations>"}}
```

list_enabled_log_categories

List non-persistent enabled radar syslog categories.

This command is available with authorization of the following access role(s):

- SUPER_USER

Radar Response Parameters

Name	Description	Type	Constraints
log_categories	Array of enabled tracked log categories	enum[]	*LOG_CATEGORIES*

Example Request

RadarIO Prompt Syntax

```
list_enabled_log_categories
```

RadarIO Script Syntax

```
client.list_enabled_log_categories()
```

JSON Syntax (as written to control port)

```
{"method": "list_enabled_log_categories", "id": 93}
```

Example Response

```
{
  "id": 93,
  "result": {"log_categories": "<Array of enabled tracked log categories>"}
}
```

query_fault

Get fault mode reason

This command is available with authorization of the following access role(s):

- USER (default)

Radar Response Parameters

Name	Description	Type	Constraints
faulted	True if the radar is in fault mode	boolean	[True, False]
reason	Reason the radar is in fault mode	string	ASCII characters
origin	Origin of the fault	string	ASCII characters
timestamp	Time the fault occurred (ms from unix epoch)	integer	'uint64_t' range

Example Request

RadarIO Prompt Syntax

```
query_fault
```

RadarIO Script Syntax

```
client.query_fault()
```

JSON Syntax (as written to control port)

```
{"method": "query_fault", "id": 120}
```

Example Response

```
{
  "id": 120,
  "result": {
    "faulted": "<True if the radar is in fault mode>",
    "reason": "<Reason the radar is in fault mode>",
    "origin": "<Origin of the fault>",
```

(continues on next page)

(continued from previous page)

```
    "timestamp": "<Time the fault occurred (ms from unix epoch)>"
  }
}
```

reboot

Reboot the radar (will lose connection after response)

This command is available with authorization of the following access role(s):

- USER (default)

Example Request

RadarIO Prompt Syntax

```
reboot
```

RadarIO Script Syntax

```
client.reboot()
```

JSON Syntax (as written to control port)

```
{"method": "reboot", "id": 106}
```

Example Response

```
{"id": 106, "result": {}}
```

set_log_level

Set the radar syslog level. NOTE: changes to log level made with this command will be in effect until the *clear_log_overrides* command is issued, or the radar is rebooted. NOTE: by design, all log messages at Warning, Error, and Critical levels are enabled regardless of this setting.

This command is available with authorization of the following access role(s):

- SUPER_USER

User Request Parameters

Name	Description	Type	Constraints
level	The log level to set	enum	*LOG_LEVELS*

Example Request

RadarIO Prompt Syntax

```
set_log_level level notice
```

RadarIO Script Syntax

```
client.set_log_level(level=notice)
```

JSON Syntax (as written to control port)

```
{"method": "set_log_level", "id": 97, "params": {"level": "notice"}}
```

Example Response

```
{"id": 97, "result": {}}
```

set_net

Set network configuration parameters (may lose connection after response)

This command is available with authorization of the following access role(s):

- USER (default)

User Request Parameters

List of Network parameter, value pairs. See the *Network Parameters* section.

Example Request

RadarIO Prompt Syntax

```
set_net address_1G 192.168.1.151 netmask_1G 255.255.254.0 gateway_1G 192.168.1.1
```

RadarIO Script Syntax

```
client.set_net(address_1G=192.168.1.151, netmask_1G=255.255.254.0, gateway_
↪1G=192.168.1.1)
```

JSON Syntax (as written to control port)

```
{
  "method": "set_net",
  "id": 110,
  "params": {
    "address_1G": "192.168.1.151",
    "netmask_1G": "255.255.254.0",
    "gateway_1G": "192.168.1.1"
  }
}
```

Example Response

```
{"id": 110, "result": {}}
```

set_sys

Set system configuration parameters

This command is available with authorization of the following access role(s):

- USER (default)

User Request Parameters

List of System parameter, value pairs. See the *System Parameters* section.

Example Request

RadarIO Prompt Syntax

```
set_sys loglevel debug
```

RadarIO Script Syntax

```
client.set_sys(loglevel=debug)
```

JSON Syntax (as written to control port)

```
{"method": "set_sys", "id": 107, "params": {"loglevel": "debug"}}
```

Example Response

```
{"id": 107, "result": {}}
```

set_systime

Set the radar's operating system time. Note: This command will return an error if the radar is using an NTP time source (see the *ntp_primary_1G*, *ntp_secondary_1G* and *ntp_tertiary_1G* parameters), or the internal GPS (see the *gps_time_disable* parameter) to keep time. See the *get_time_info* command to view the active time source.

This command is available with authorization of the following access role(s):

- USER (default)

User Request Parameters

Name	Description	Type	Constraints
time_unix_ns	Time to set, in nanoseconds from Unix epoch	integer	'int64_t' range

Example Request

RadarIO Prompt Syntax

```
set_systime time_unix_ns 8757775406709268826
```

RadarIO Script Syntax

```
client.set_systime(time_unix_ns=8757775406709268826)
```

JSON Syntax (as written to control port)

```
{
  "method": "set_systime",
  "id": 101,
  "params": {"time_unix_ns": 8757775406709268826}
}
```

Example Response

```
{"id": 101, "result": {}}
```

3.5 Radar App Commands

Commands for controlling and configuring the Radar App:

- *clear_zone_mask*
- *configure_mission*
- *get*
- *get_ins_data*
- *get_mission*
- *get_radar_kinematics*
- *get_radar_kinematics_ecef*
- *get_radar_kinematics_geodetic*
- *get_radarapp_deltas*
- *get_reference_frame_offsets*
- *get_reference_kinematics_ecef*
- *get_reference_kinematics_geodetic*
- *get_zone_mask*
- *insert_zone_mask*
- *license_mission*
- *list_missions*
- *mode_set_start*
- *mode_set_stop*
- *remove_zone_mask*
- *reset_radar_params*
- *set*
- *set_radar_kinematics*
- *set_reference_frame_offsets*
- *set_reference_kinematics_ecef*
- *set_reference_kinematics_geodetic*
- *spectrum_test_start*
- *spectrum_test_stop*
- *track_focus_start*

- *track_focus_stop*
- *track_handoff*

clear_zone_mask

Removes all existing measurement based zone masks.

This command is available with authorization of the following access role(s):

- USER (default)

Example Request

RadarIO Prompt Syntax

```
clear_zone_mask
```

RadarIO Script Syntax

```
client.clear_zone_mask()
```

JSON Syntax (as written to control port)

```
{"method": "clear_zone_mask", "id": 88}
```

Example Response

```
{"id": 88, "result": {}}
```

configure_mission

Configure system to execute specified mission set. This will reset all volatile RadarApp parameters to their default relative to the active mission. The system must be in the service state to configure mission set.

This command is available with authorization of the following access role(s):

- USER (default)

User Request Parameters

Name	Description	Type	Constraints
mission	Configurable mission sets. NOTE: The mission set should only be updated through the configure_mission command.	enum	• C-UAS_1 • C-UAS_2 • DISMOUNT • COASTAL_C-UAS

Example Request

RadarIO Prompt Syntax

```
configure_mission mission C-UAS_1
```

RadarIO Script Syntax

```
client.configure_mission(mission=C-UAS_1)
```

JSON Syntax (as written to control port)

```
{"method": "configure_mission", "id": 81, "params": {"mission": "C-UAS_1"}}
```

Example Response

```
{"id": 81, "result": {}}
```

get

Get Radar App configuration parameters

This command is available with authorization of the following access role(s):

- USER (default)

User Request Parameters

Name	Description	Type	Constraints
retrieve	**Optional** Parameters to retrieve	string[]	ASCII characters Maximum 256 elements

Radar Response Parameters

Dictionary of specified radar parameter values.

Example Request

RadarIO Prompt Syntax

```
get retrieve [search_azmin, search_azmax]
```

RadarIO Script Syntax

```
client.get(retrieve=['search_azmin', 'search_azmax'])
```

JSON Syntax (as written to control port)

```
{
  "method": "get",
  "id": 62,
  "params": {"retrieve": ["search_azmin", "search_azmax"]}
}
```

Example Response


```
{"id": 62, "result": {"search_azmin": -50, "search_azmax": 50}}
```

`get_ins_data`

Get the position and orientation of the radar based on its internal sensors. A user can use altitude_type as an optional input parameter to specify the vertical datum used for the returned altitude. If no altitude_type is provided, orthometric is used by default.

This command is available with authorization of the following access role(s):

- USER (default)

User Request Parameters

Name	Description	Type	Constraints
altitude_type	**Optional** Defines the vertical datum used for the given altitude. The ellipsoidal height/altitude refers to the height above the ellipsoid. Echoshield assumes use of the WGS84 ellipsoid. Orthometric altitude is the height above the geoid, which represents mean sea level and is determined by Earth's gravity. Echoshield assumes use of the EGM96 geoid, but should not be sensitive to other geoid models. It is important to explicitly define whether a given altitude is the height above the ellipsoid or geoid as these can differ by as much as ~90 meters depending on where you are in the world.	enum	• ellipsoidal • orthometric

Radar Response Parameters

Name	Description	Type	Constraints
pitch_roll_mature	This boolean is true if the INS has been collecting IMU data for a sufficient duration. If this is false, then the pitch and roll filter does not have enough data to output a stable solution, and the pitch-back and tilt outputs should be considered invalid.	boolean	[True, False]
heading_mature	This boolean is true if the INS has been collecting GPS data for a sufficient duration. If this is false, then the heading filter does not have enough data to output a stable solution, and the heading outputs should be considered invalid.	boolean	[True, False]
gps_fix_mode	Mode of GPS fix. The GPS must have 3D lock for INS position and heading fields to be considered valid. If the GPS has a 2D lock, the altitude and heading should be considered invalid. If there is no fix, the latitude, longitude, altitude and heading reported by the INS should be considered invalid.	enum	• NOT_SEEN • NO_FIX • 2D • 3D
gps_heading_valid	The internal GPS sensor's estimate of if its heading solution is valid. If this is invalid, the internal INS heading solution should be considered invalid.	boolean	[True, False]
latitude_deg	WGS84 latitude position in degrees	number	$-90 \leq x \leq 90.0$
longitude_deg	WGS84 longitude position in degrees	number	$-180.0 \leq x \leq 180.0$

continues on next page

Table 30 – continued from previous page

Name	Description	Type	Constraints
altitude_m	The altitude relative to the specified vertical datum. The vertical datum is explicitly defined by the altitude_type field.	number	$-1000.0 \leq x \leq 1000000$
altitude_type	Defines the vertical datum used for the given altitude. The ellipsoidal height/altitude refers to the height above the ellipsoid. Echoshield assumes use of the WGS84 ellipsoid. Orthometric altitude is the height above the geoid, which represents mean sea level and is determined by Earth's gravity. Echoshield assumes use of the EGM96 geoid, but should not be sensitive to other geoid models. It is important to explicitly define whether a given altitude is the height above the ellipsoid or geoid as these can differ by as much as ~90 meters depending on where you are in the world.	enum	• ellipsoidal • orthometric
heading_deg	Number of degrees clockwise from north that the radar's z-axis (broadside) is pointed.	number	$0.0 \leq x \leq 360.0$
pitchback_deg	Number of degrees above the horizon (the radar's level plane) that radar z-axis (broadside) is pointing.	number	$-180.0 \leq x \leq 180.0$
tilt_deg	The left-handed rotation degrees around radar z-axis (broadside).	number	$-180.0 \leq x \leq 180.0$
quaternion_ecef_w	W quaternion orientation in ECEF frame	number	'double' range
quaternion_ecef_x	X quaternion orientation in ECEF frame	number	'double' range

continues on next page

Table 30 – continued from previous page

Name	Description	Type	Constraints
quaternion_ecef_y	Y quaternion orientation in ECEF frame	number	'double' range
quaternion_ecef_z	Z quaternion orientation in ECEF frame	number	'double' range
enhanced_accelerometer_cal	True if this unit has undergone enhanced accelerometer calibration	boolean	[True, False]

Example Request

RadarIO Prompt Syntax

```
get_ins_data altitude_type orthometric
```

RadarIO Script Syntax

```
client.get_ins_data(altitude_type=orthometric)
```

JSON Syntax (as written to control port)

```
{
  "method": "get_ins_data", "id": 80, "params": {"altitude_type": "orthometric"}
}
```

Example Response

```
{
  "id": 80,
  "result": {
    "pitch_roll_mature": "<This boolean is true if the INS has been collecting␣
↪IMU data for a sufficient duration. If this is false, then the pitch and roll␣
↪filter does not have enough data to output a stable solution, and the␣
↪pitchback and tilt outputs should be considered invalid. >",
    "heading_mature": "<This boolean is true if the INS has been collecting GPS␣
↪data for a sufficient duration. If this is false, then the heading filter does␣
↪not have enough data to output a stable solution, and the heading outputs␣
↪should be considered invalid. >",
    "gps_fix_mode": "<Mode of GPS fix. The GPS must have 3D lock for INS␣
↪position and heading fields to be considered valid. If the GPS has a 2D lock,␣
```

(continues on next page)

(continued from previous page)

```
↪the altitude and heading should be considered invalid. If there is no fix, the␣
↪latitude, longitude, altitude and heading reported by the INS should be␣
↪considered invalid.>",
    "gps_heading_valid": "<The internal GPS sensor's estimate of if its heading␣
↪solution is valid. If this is invalid, the internal INS heading solution␣
↪should be considered invalid. >",
    "latitude_deg": "<WGS84 latitude position in degrees>",
    "longitude_deg": "<WGS84 longitude position in degrees>",
    "altitude_m": "<The altitude relative to the specified vertical datum.  The␣
↪vertical datum is explicitly defined by the altitude_type field.>",
    "altitude_type": "<Defines the vertical datum used for the given altitude.␣
↪The ellipsoidal height/altitude refers to the height above the ellipsoid. ␣
↪Echoshield assumes use of the WGS84 ellipsoid.  Orthometric altitude is the␣
↪height above the geoid, which represents mean sea level and is determined by␣
↪Earth's gravity.  Echoshield assumes use of the EGM96 geoid, but should not be␣
↪sensitive to other geoid models.  It is important to explicitly define whether␣
↪a given altitude is the height above the ellipsoid or geoid as these can␣
↪differ by as much as ~90 meters depending on where you are in the world.>",
    "heading_deg": "<Number of degrees clockwise from north that the radar's z-
↪axis (broadside) is pointed. >",
    "pitchback_deg": "<Number of degrees above the horizon (the radar's level␣
↪plane) that radar z-axis (broadside) is pointing. >",
    "tilt_deg": "<The left-handed rotation degrees around radar z-axis␣
↪(broadside). >",
    "quaternion_ecef_w": "<W quaternion orientation in ECEF frame>",
    "quaternion_ecef_x": "<X quaternion orientation in ECEF frame>",
    "quaternion_ecef_y": "<Y quaternion orientation in ECEF frame>",
    "quaternion_ecef_z": "<Z quaternion orientation in ECEF frame>",
    "enhanced_accelerometer_cal": "<True if this unit has undergone enhanced␣
↪accelerometer calibration>"
  }
}
```

get_mission

Get current mission set configuration.

This command is available with authorization of the following access role(s):

- USER (default)

Radar Response Parameters

Name	Description	Type	Constraints
mission	Configurable mission sets. NOTE: The mission set should only be updated through the configure_mission command.	enum	• C-UAS_1 • C-UAS_2 • DISMOUNT • COASTAL_C-UAS

Example Request

RadarIO Prompt Syntax

```
get_mission
```

RadarIO Script Syntax

```
client.get_mission()
```

JSON Syntax (as written to control port)

```
{"method": "get_mission", "id": 82}
```

Example Response

```
{
  "id": 82,
  "result": {
    "mission": "<Configurable mission sets.  NOTE: The mission set should only␣
↪be updated through the configure_mission command.>"
  }
}
```

get_radar_kinematics

Get user-set geodetic position and heading, pitch, and tilt of the radar kinematics. A user can use altitude_type as an optional input parameter to specify the vertical datum used for the returned altitude. If no altitude_type is provided, orthometric is used by default.

This command is available with authorization of the following access role(s):

- USER (default)

User Request Parameters

Name	Description	Type	Constraints
altitude_type	**Optional** Defines the vertical datum used for the given altitude. The ellipsoidal height/altitude refers to the height above the ellipsoid. Echoshield assumes use of the WGS84 ellipsoid. Orthometric altitude is the height above the geoid, which represents mean sea level and is determined by Earth's gravity. Echoshield assumes use of the EGM96 geoid, but should not be sensitive to other geoid models. It is important to explicitly define whether a given altitude is the height above the ellipsoid or geoid as these can differ by as much as ~90 meters depending on where you are in the world.	enum	• ellipsoidal • orthometric

Radar Response Parameters

Name	Description	Type	Constraints
latitude_deg	WGS84 latitude position in degrees	number	$-90.0 \leq x \leq 90.0$
longitude_deg	WGS84 longitude position in degrees	number	$-180.0 \leq x \leq 180.0$
altitude_m	The altitude relative to the specified vertical datum. The vertical datum is explicitly defined by the altitude_type field.	number	$-1000.0 \leq x \leq 1000000$
heading_deg	Number of degrees clockwise from north that the radar's z-axis (broadside) is pointed	number	$0.0 \leq x \leq 360.0$
pitchback_deg	Number of degrees above the horizon (the radar's level plane) that radar z-axis (broadside) is pointing	number	$-180.0 \leq x \leq 180.0$
tilt_deg	The left-handed rotation degrees around radar z-axis (broadside)	number	$-180.0 \leq x \leq 180.0$

continues on next page

Table 33 – continued from previous page

Name	Description	Type	Constraints
altitude_type	Defines the vertical datum used for the given altitude. The ellipsoidal height/altitude refers to the height above the ellipsoid. Echoshield assumes use of the WGS84 ellipsoid. Orthometric altitude is the height above the geoid, which represents mean sea level and is determined by Earth's gravity. Echoshield assumes use of the EGM96 geoid, but should not be sensitive to other geoid models. It is important to explicitly define whether a given altitude is the height above the ellipsoid or geoid as these can differ by as much as ~90 meters depending on where you are in the world.	enum	• ellipsoidal • orthometric
kinematics_id	A unique ID that is incremented each time the kinematics is updated	integer	'uint64_t' range
kinematics_sensor_agreement	How closely the user-set kinematics agrees with the internal INS solution	string	ASCII characters

Example Request

RadarIO Prompt Syntax

```
get_radar_kinematics altitude_type ellipsoidal
```

RadarIO Script Syntax

```
client.get_radar_kinematics(altitude_type=ellipsoidal)
```

JSON Syntax (as written to control port)

```
{
  "method": "get_radar_kinematics",
  "id": 75,
  "params": {"altitude_type": "ellipsoidal"}
}
```

Example Response

```
{
  "id": 75,
  "result": {
    "latitude_deg": "<WGS84 latitude position in degrees>",
    "longitude_deg": "<WGS84 longitude position in degrees>",
    "altitude_m": "<The altitude relative to the specified vertical datum.  The␣
↪vertical datum is explicitly defined by the altitude_type field.>",
    "heading_deg": "<Number of degrees clockwise from north that the radar's z-
↪axis (broadside) is pointed >",
    "pitchback_deg": "<Number of degrees above the horizon (the radar's level␣
↪plane) that radar z-axis (broadside) is pointing>",
    "tilt_deg": "<The left-handed rotation degrees around radar z-axis␣
↪(broadside)>",
    "altitude_type": "<Defines the vertical datum used for the given altitude.␣
↪The ellipsoidal height/altitude refers to the height above the ellipsoid. ␣
↪Echoshield assumes use of the WGS84 ellipsoid.  Orthometric altitude is the␣
↪height above the geoid, which represents mean sea level and is determined by␣
↪Earth's gravity.  Echoshield assumes use of the EGM96 geoid, but should not be␣
↪sensitive to other geoid models.  It is important to explicitly define whether␣
↪a given altitude is the height above the ellipsoid or geoid as these can␣
↪differ by as much as ~90 meters depending on where you are in the world.>",
    "kinematics_id": "<A unique ID that is incremented each time the kinematics␣
↪is updated>",
    "kinematics_sensor_agreement": "<How closely the user-set kinematics agrees␣
↪with the internal INS solution>"
  }
}
```

`get_radar_kinematics_ecef`

Get user-set ECEF-based radar frame kinematics.

This command is available with authorization of the following access role(s):

- USER (default)

Radar Response Parameters

Name	Description	Type	Constraints
position_ecef_x	X position in ECEF frame in meters	number	$-100000000 \leq x \leq 100000000$
position_ecef_y	Y position in ECEF frame in meters	number	$-100000000 \leq x \leq 100000000$
position_ecef_z	Z position in ECEF frame in meters	number	$-100000000 \leq x \leq 100000000$
quaternion_ecef_w	W quaternion orientation in ECEF frame	number	'double' range
quaternion_ecef_x	X quaternion orientation in ECEF frame	number	'double' range
quaternion_ecef_y	Y quaternion orientation in ECEF frame	number	'double' range
quaternion_ecef_z	Z quaternion orientation in ECEF frame	number	'double' range
velocity_ecef_x	X velocity in ECEF frame in meters per second	number	'double' range
velocity_ecef_y	Y velocity in ECEF frame in meters per second	number	'double' range
velocity_ecef_z	Z velocity in ECEF frame in meters per second	number	'double' range
rotation_rate_ecef_x	X angular velocity in ECEF frame in radians per second	number	'double' range
rotation_rate_ecef_y	Y angular velocity in ECEF frame in radians per second	number	'double' range
rotation_rate_ecef_z	Z angular velocity in ECEF frame in radians per second	number	'double' range
kinematics_id	A unique ID that is incremented each time the kinematics is updated.	integer	'uint64_t' range
kinematics_sensor_agreement	How closely the user-set kinematics agrees with the internal INS solution.	string	ASCII characters

continues on next page

Table 34 – continued from previous page

Name	Description	Type	Constraints

Example Request

RadarIO Prompt Syntax

```
get_radar_kinematics_ecef
```

RadarIO Script Syntax

```
client.get_radar_kinematics_ecef()
```

JSON Syntax (as written to control port)

```
{"method": "get_radar_kinematics_ecef", "id": 78}
```

Example Response

```
{
  "id": 78,
  "result": {
    "position_ecef_x": "<X position in ECEF frame in meters>",
    "position_ecef_y": "<Y position in ECEF frame in meters>",
    "position_ecef_z": "<Z position in ECEF frame in meters>",
    "quaternion_ecef_w": "<W quaternion orientation in ECEF frame>",
    "quaternion_ecef_x": "<X quaternion orientation in ECEF frame>",
    "quaternion_ecef_y": "<Y quaternion orientation in ECEF frame>",
    "quaternion_ecef_z": "<Z quaternion orientation in ECEF frame>",
    "velocity_ecef_x": "<X velocity in ECEF frame in meters per second>",
    "velocity_ecef_y": "<Y velocity in ECEF frame in meters per second>",
    "velocity_ecef_z": "<Z velocity in ECEF frame in meters per second>",
    "rotation_rate_ecef_x": "<X angular velocity in ECEF frame in radians per␣
↪second>",
    "rotation_rate_ecef_y": "<Y angular velocity in ECEF frame in radians per␣
↪second>",
    "rotation_rate_ecef_z": "<Z angular velocity in ECEF frame in radians per␣
↪second>",
    "kinematics_id": "<A unique ID that is incremented each time the kinematics␣
↪is updated. >",
    "kinematics_sensor_agreement": "<How closely the user-set kinematics agrees␣
↪with the internal INS solution. >"
```

(continues on next page)

(continued from previous page)

```
  }
}
```

get_radar_kinematics_geodetic

Get user-set geodetic radar kinematics. A user can use altitude_type as an optional input parameter to specify the vertical datum used for the returned altitude. If no altitude_type is provided, orthometric is used by default.

This command is available with authorization of the following access role(s):

- USER (default)

User Request Parameters

Name	Description	Type	Constraints
altitude_type	**Optional** Defines the vertical datum used for the given altitude. The ellipsoidal height/altitude refers to the height above the ellipsoid. Echoshield assumes use of the WGS84 ellipsoid. Orthometric altitude is the height above the geoid, which represents mean sea level and is determined by Earth's gravity. Echoshield assumes use of the EGM96 geoid, but should not be sensitive to other geoid models. It is important to explicitly define whether a given altitude is the height above the ellipsoid or geoid as these can differ by as much as ~90 meters depending on where you are in the world.	enum	• ellipsoidal • orthometric

Radar Response Parameters

Name	Description	Type	Constraints
latitude_deg	WGS84 latitude position in degrees	number	$-90.0 \leq x \leq 90.0$
longitude_deg	WGS84 longitude position in degrees	number	$-180.0 \leq x \leq 180.0$
altitude_m	The altitude relative to the specified vertical datum. The vertical datum is explicitly defined by the altitude_type field.	number	$-1000.0 \leq x \leq 1000000$
yaw_enu_deg	The right-handed yaw of the radar frame around the ENU frame's z-axis, in degrees. This is the first rotation applied in the Yaw-Pitch-Roll sequence, meaning the rotation is applied relative to the ENU frame	number	$-360.0 \leq x \leq 360.0$
pitch_enu_deg	The right-handed pitch of the radar frame around the yaw-rotated frame's y-axis, in degrees. This is the second rotation in the Yaw-Pitch-Roll sequence, meaning it is applied after the yaw rotation and is relative to the yaw-rotated frame.	number	$-360.0 \leq x \leq 360.0$
roll_enu_deg	The right-handed roll of the radar frame around the pitch-and-roll-rotated frame's x-axis, in degrees. This is the final rotation in the Yaw-Pitch-Roll sequence, meaning it is applied after both yaw and pitch rotations, relative to the already-rotated frame.	number	$-360.0 \leq x \leq 360.0$

continues on next page

Table 36 – continued from previous page

Name	Description	Type	Constraints
altitude_type	Defines the vertical datum used for the given altitude. The ellipsoidal height/altitude refers to the height above the ellipsoid. Echoshield assumes use of the WGS84 ellipsoid. Orthometric altitude is the height above the geoid, which represents mean sea level and is determined by Earth's gravity. Echoshield assumes use of the EGM96 geoid, but should not be sensitive to other geoid models. It is important to explicitly define whether a given altitude is the height above the ellipsoid or geoid as these can differ by as much as ~90 meters depending on where you are in the world.	enum	• ellipsoidal • orthometric
kinematics_id	A unique ID that is incremented each time the kinematics is updated.	integer	'uint64_t' range
kinematics_sensor_agreement	How closely the user-set kinematics agrees with the internal INS solution.	string	ASCII characters

Example Request

RadarIO Prompt Syntax

```
get_radar_kinematics_geodetic altitude_type orthometric
```

RadarIO Script Syntax

```
client.get_radar_kinematics_geodetic(altitude_type=orthometric)
```

JSON Syntax (as written to control port)

```
{
  "method": "get_radar_kinematics_geodetic",
  "id": 79,
  "params": {"altitude_type": "orthometric"}
}
```

Example Response

```
{
  "id": 79,
  "result": {
    "latitude_deg": "<WGS84 latitude position in degrees>",
    "longitude_deg": "<WGS84 longitude position in degrees>",
    "altitude_m": "<The altitude relative to the specified vertical datum.  The␣
↪vertical datum is explicitly defined by the altitude_type field.>",
    "yaw_enu_deg": "<The right-handed yaw of the radar frame around the ENU frame
↪'s z-axis, in degrees. This is the first rotation applied in the Yaw-Pitch-
↪Roll sequence, meaning the rotation is applied relative to the ENU frame>",
    "pitch_enu_deg": "<The right-handed pitch of the radar frame around the yaw-
↪rotated frame's y-axis, in degrees. This is the second rotation in the Yaw-
↪Pitch-Roll sequence, meaning it is applied after the yaw rotation and is␣
↪relative to the yaw-rotated frame.>",
    "roll_enu_deg": "<The right-handed roll of the radar frame around the pitch-
↪and-roll-rotated frame's x-axis, in degrees. This is the final rotation in the␣
↪Yaw-Pitch-Roll sequence, meaning it is applied after both yaw and pitch␣
↪rotations, relative to the already-rotated frame.>",
    "altitude_type": "<Defines the vertical datum used for the given altitude.␣
↪The ellipsoidal height/altitude refers to the height above the ellipsoid. ␣
↪Echoshield assumes use of the WGS84 ellipsoid.  Orthometric altitude is the␣
↪height above the geoid, which represents mean sea level and is determined by␣
↪Earth's gravity.  Echoshield assumes use of the EGM96 geoid, but should not be␣
↪sensitive to other geoid models.  It is important to explicitly define whether␣
↪a given altitude is the height above the ellipsoid or geoid as these can␣
↪differ by as much as ~90 meters depending on where you are in the world.>",
    "kinematics_id": "<A unique ID that is incremented each time the kinematics␣
↪is updated. >",
    "kinematics_sensor_agreement": "<How closely the user-set kinematics agrees␣
↪with the internal INS solution. >"
  }
}
```

get_radarapp_deltas

Get a list of all radar app configuration parameters that are not set to their default values.

This command is available with authorization of the following access role(s):

- USER (default)

Radar Response Parameters

Dictionary of parameters that are not set at their default values.

Example Request

RadarIO Prompt Syntax

```
get_radarapp_deltas
```

RadarIO Script Syntax

```
client.get_radarapp_deltas()
```

JSON Syntax (as written to control port)

```
{"method": "get_radarapp_deltas", "id": 63}
```

Example Response

```
{"id": 63, "result": {"search_azmax": -50, "search_azmin": 50}}
```

`get_reference_frame_offsets`

Retrieve the offsets of the radar frame with respect to the reference frame. If the offsets have not yet been set, the 'offsets_set' flag will be false, and the values will be all zero.

This command is available with authorization of the following access role(s):

- USER (default)

Radar Response Parameters

Name	Description	Type	Constraints
position_relative_to _reference_x	The position of the radar frame on the reference-frame's x-axis, in meters.	number	$-1000000 \leq x \leq 1000000$
position_relative_to _reference_y	The position of the radar frame on the reference-frame's y-axis, in meters.	number	$-1000000 \leq x \leq 1000000$
position_relative_to _reference_z	The position of the radar frame on the reference-frame's z-axis, in meters.	number	$-1000000 \leq x \leq 1000000$
rotation_relative_to _reference_yaw	The right-handed yaw of the radar frame around the reference-frame's z-axis, in degrees. This is the first rotation applied in the Yaw-Pitch-Roll sequence, meaning the rotation is applied relative to the original reference-frame.	number	$-360.0 \leq x \leq 360.0$
rotation_relative_to _reference_pitch	The right-handed pitch of the radar frame around the yaw-rotated frame's y-axis, in degrees. This is the second rotation in the Yaw-Pitch-Roll sequence, meaning it is applied after the yaw rotation and is relative to the yaw-rotated frame.	number	$-360.0 \leq x \leq 360.0$

continues on next page

Table 37 – continued from previous page

Name	Description	Type	Constraints
rotation_relative_to _reference_roll	The right-handed roll of the radar frame around the yaw-and-pitch-rotated frame's x-axis, in degrees. This is the final rotation in the Yaw-Pitch-Roll sequence, meaning it is applied after both yaw and pitch rotations, relative to the already-rotated frame.	number	$-360.0 \leq x \leq 360.0$
offsets_set	Flag that is true if the user has previously set the reference frame offsets. The reference frame offsets must be set prior to setting the reference frame kinematics.	boolean	[True, False]

Example Request

RadarIO Prompt Syntax

```
get_reference_frame_offsets
```

RadarIO Script Syntax

```
client.get_reference_frame_offsets()
```

JSON Syntax (as written to control port)

```
{"method": "get_reference_frame_offsets", "id": 72}
```

Example Response

```
{
  "id": 72,
  "result": {
    "position_relative_to_reference_x": "<The position of the radar frame on the␣
↪reference-frame's x-axis, in meters. >",
    "position_relative_to_reference_y": "<The position of the radar frame on the␣
↪reference-frame's y-axis, in meters. >",
```

(continues on next page)

(continued from previous page)

```
    "position_relative_to_reference_z": "<The position of the radar frame on the␣
↪reference-frame's z-axis, in meters. >",
    "rotation_relative_to_reference_yaw": "<The right-handed yaw of the radar␣
↪frame around the reference-frame's z-axis, in degrees. This is the first␣
↪rotation applied in the Yaw-Pitch-Roll sequence, meaning the rotation is␣
↪applied relative to the original reference-frame.>",
    "rotation_relative_to_reference_pitch": "<The right-handed pitch of the␣
↪radar frame around the yaw-rotated frame's y-axis, in degrees. This is the␣
↪second rotation in the Yaw-Pitch-Roll sequence, meaning it is applied after␣
↪the yaw rotation and is relative to the yaw-rotated frame.>",
    "rotation_relative_to_reference_roll": "<The right-handed roll of the radar␣
↪frame around the yaw-and-pitch-rotated frame's x-axis, in degrees. This is the␣
↪final rotation in the Yaw-Pitch-Roll sequence, meaning it is applied after␣
↪both yaw and pitch rotations, relative to the already-rotated frame.>",
    "offsets_set": "<Flag that is true if the user has previously set the␣
↪reference frame offsets. The reference frame offsets must be set prior to␣
↪setting the reference frame kinematics.>"
  }
}
```

get_reference_kinematics_ecef

Get user-set ECEF-based reference frame kinematics.

This command is available with authorization of the following access role(s):

- USER (default)

Radar Response Parameters

Name	Description	Type	Constraints
position_ecef_x	X position in ECEF frame in meters	number	$-100000000 \leq x \leq 100000000$
position_ecef_y	Y position in ECEF frame in meters	number	$-100000000 \leq x \leq 100000000$
position_ecef_z	Z position in ECEF frame in meters	number	$-100000000 \leq x \leq 100000000$
quaternion_ecef_w	W quaternion orientation in ECEF frame	number	'double' range
quaternion_ecef_x	X quaternion orientation in ECEF frame	number	'double' range
quaternion_ecef_y	Y quaternion orientation in ECEF frame	number	'double' range
quaternion_ecef_z	Z quaternion orientation in ECEF frame	number	'double' range
velocity_ecef_x	X velocity in ECEF frame in meters per second	number	'double' range
velocity_ecef_y	Y velocity in ECEF frame in meters per second	number	'double' range
velocity_ecef_z	Z velocity in ECEF frame in meters per second	number	'double' range
rotation_rate_ecef_x	X angular velocity in ECEF frame in radians per second	number	'double' range
rotation_rate_ecef_y	Y angular velocity in ECEF frame in radians per second	number	'double' range
rotation_rate_ecef_z	Z angular velocity in ECEF frame in radians per second	number	'double' range
kinematics_id	A unique ID that is incremented each time the kinematics is updated.	integer	'uint64_t' range
kinematics_sensor_agreement	How closely the user-set kinematics agrees with the internal INS solution.	string	ASCII characters

continues on next page

Table 38 – continued from previous page

Name	Description	Type	Constraints

Example Request

RadarIO Prompt Syntax

```
get_reference_kinematics_ecef
```

RadarIO Script Syntax

```
client.get_reference_kinematics_ecef()
```

JSON Syntax (as written to control port)

```
{"method": "get_reference_kinematics_ecef", "id": 76}
```

Example Response

```
{
  "id": 76,
  "result": {
    "position_ecef_x": "<X position in ECEF frame in meters>",
    "position_ecef_y": "<Y position in ECEF frame in meters>",
    "position_ecef_z": "<Z position in ECEF frame in meters>",
    "quaternion_ecef_w": "<W quaternion orientation in ECEF frame>",
    "quaternion_ecef_x": "<X quaternion orientation in ECEF frame>",
    "quaternion_ecef_y": "<Y quaternion orientation in ECEF frame>",
    "quaternion_ecef_z": "<Z quaternion orientation in ECEF frame>",
    "velocity_ecef_x": "<X velocity in ECEF frame in meters per second>",
    "velocity_ecef_y": "<Y velocity in ECEF frame in meters per second>",
    "velocity_ecef_z": "<Z velocity in ECEF frame in meters per second>",
    "rotation_rate_ecef_x": "<X angular velocity in ECEF frame in radians per␣
↪second>",
    "rotation_rate_ecef_y": "<Y angular velocity in ECEF frame in radians per␣
↪second>",
    "rotation_rate_ecef_z": "<Z angular velocity in ECEF frame in radians per␣
↪second>",
    "kinematics_id": "<A unique ID that is incremented each time the kinematics␣
↪is updated. >",
    "kinematics_sensor_agreement": "<How closely the user-set kinematics agrees␣
↪with the internal INS solution. >"
```

(continues on next page)

(continued from previous page)

```
  }
}
```

get_reference_kinematics_geodetic

Get user-set geodetic- and euler-based reference frame kinematics values. A user can use altitude_type as an optional input parameter to specify the vertical datum used for the returned altitude. If no altitude_type is provided, orthometric is used by default.

This command is available with authorization of the following access role(s):

- USER (default)

User Request Parameters

Name	Description	Type	Constraints
altitude_type	**Optional** Defines the vertical datum used for the given altitude. The ellipsoidal height/altitude refers to the height above the ellipsoid. Echoshield assumes use of the WGS84 ellipsoid. Orthometric altitude is the height above the geoid, which represents mean sea level and is determined by Earth's gravity. Echoshield assumes use of the EGM96 geoid, but should not be sensitive to other geoid models. It is important to explicitly define whether a given altitude is the height above the ellipsoid or geoid as these can differ by as much as ~90 meters depending on where you are in the world.	enum	• ellipsoidal • orthometric

Radar Response Parameters

Name	Description	Type	Constraints
latitude_deg	WGS84 latitude position in degrees	number	$-90.0 \leq x \leq 90.0$
longitude_deg	WGS84 longitude position in degrees	number	$-180.0 \leq x \leq 180.0$
altitude_m	The altitude relative to the specified vertical datum. The vertical datum is explicitly defined by the altitude_type field.	number	$-1000.0 \leq x \leq 1000000$
yaw_enu_deg	The right-handed yaw of the radar frame around the ENU frame's z-axis, in degrees. This is the first rotation applied in the Yaw-Pitch-Roll sequence, meaning the rotation is applied relative to the ENU frame	number	$-360.0 \leq x \leq 360.0$
pitch_enu_deg	The right-handed pitch of the radar frame around the yaw-rotated frame's y-axis, in degrees. This is the second rotation in the Yaw-Pitch-Roll sequence, meaning it is applied after the yaw rotation and is relative to the yaw-rotated frame.	number	$-360.0 \leq x \leq 360.0$
roll_enu_deg	The right-handed roll of the radar frame around the pitch-and-roll-rotated frame's x-axis, in degrees. This is the final rotation in the Yaw-Pitch-Roll sequence, meaning it is applied after both yaw and pitch rotations, relative to the already-rotated frame.	number	$-360.0 \leq x \leq 360.0$

continues on next page

Table 40 – continued from previous page

Name	Description	Type	Constraints
altitude_type	Defines the vertical datum used for the given altitude. The ellipsoidal height/altitude refers to the height above the ellipsoid. Echoshield assumes use of the WGS84 ellipsoid. Orthometric altitude is the height above the geoid, which represents mean sea level and is determined by Earth's gravity. Echoshield assumes use of the EGM96 geoid, but should not be sensitive to other geoid models. It is important to explicitly define whether a given altitude is the height above the ellipsoid or geoid as these can differ by as much as ~90 meters depending on where you are in the world.	enum	• ellipsoidal • orthometric
kinematics_id	A unique ID that is incremented each time the kinematics is updated.	integer	'uint64_t' range
kinematics_sensor_agreement	How closely the user-set kinematics agrees with the internal INS solution.	string	ASCII characters

Example Request

RadarIO Prompt Syntax

```
get_reference_kinematics_geodetic altitude_type orthometric
```

RadarIO Script Syntax

```
client.get_reference_kinematics_geodetic(altitude_type=orthometric)
```

JSON Syntax (as written to control port)

```
{
  "method": "get_reference_kinematics_geodetic",
  "id": 77,
  "params": {"altitude_type": "orthometric"}
}
```

Example Response

```
{
  "id": 77,
  "result": {
    "latitude_deg": "<WGS84 latitude position in degrees>",
    "longitude_deg": "<WGS84 longitude position in degrees>",
    "altitude_m": "<The altitude relative to the specified vertical datum.  The␣
↪vertical datum is explicitly defined by the altitude_type field.>",
    "yaw_enu_deg": "<The right-handed yaw of the radar frame around the ENU frame
↪'s z-axis, in degrees. This is the first rotation applied in the Yaw-Pitch-
↪Roll sequence, meaning the rotation is applied relative to the ENU frame>",
    "pitch_enu_deg": "<The right-handed pitch of the radar frame around the yaw-
↪rotated frame's y-axis, in degrees. This is the second rotation in the Yaw-
↪Pitch-Roll sequence, meaning it is applied after the yaw rotation and is␣
↪relative to the yaw-rotated frame.>",
    "roll_enu_deg": "<The right-handed roll of the radar frame around the pitch-
↪and-roll-rotated frame's x-axis, in degrees. This is the final rotation in the␣
↪Yaw-Pitch-Roll sequence, meaning it is applied after both yaw and pitch␣
↪rotations, relative to the already-rotated frame.>",
    "altitude_type": "<Defines the vertical datum used for the given altitude.␣
↪The ellipsoidal height/altitude refers to the height above the ellipsoid. ␣
↪Echoshield assumes use of the WGS84 ellipsoid.  Orthometric altitude is the␣
↪height above the geoid, which represents mean sea level and is determined by␣
↪Earth's gravity.  Echoshield assumes use of the EGM96 geoid, but should not be␣
↪sensitive to other geoid models.  It is important to explicitly define whether␣
↪a given altitude is the height above the ellipsoid or geoid as these can␣
↪differ by as much as ~90 meters depending on where you are in the world.>",
    "kinematics_id": "<A unique ID that is incremented each time the kinematics␣
↪is updated. >",
    "kinematics_sensor_agreement": "<How closely the user-set kinematics agrees␣
↪with the internal INS solution. >"
  }
}
```

get_zone_mask

Get currently configured zone masks.

This command is available with authorization of the following access role(s):

- USER (default)

Radar Response Parameters

Name	Description	Type	Constraints
list	Contains a list of measurement zone masks	*Data_zon* []	Maximum 32 elements

Data_zone_mask

Name	Description	Type	Constraints
id	Unique identifier for the zone mask. If no ID is provided, the system will provide a unique ID.	integer	$0 \leq x \leq 31$
coordinates	Contains a series of geodetic coordinates used to define a polygon. The points must be ordered so that the defined polygon is not self-intersecting. The polygon defines the zone mask region.	*Data_lat_* []	Maximum 32 elements
agl_max	Max height above ground in meters at which to apply the zone mask. If no value is provided for agl_max, agl_max = max value.	number	$0 \leq x \leq 1000000.0$

Data_lat_long_point

Name	Description	Type	Constraints
latitude_deg	WGS84 latitude position in degrees	number	$-90.0 \leq x \leq 90.0$
longitude_deg	WGS84 longitude position in degrees	number	$-180.0 \leq x \leq 180.0$

Example Request

RadarIO Prompt Syntax

```
get_zone_mask
```

RadarIO Script Syntax

```
client.get_zone_mask()
```

JSON Syntax (as written to control port)

```
{"method": "get_zone_mask", "id": 85}
```

Example Response

```
{"id": 85, "result": {"list": "<Contains a list of measurement zone masks>"}}
```

insert_zone_mask

Add a list of measurement based zone masks. These masks are added to any existing measurement-based zone masks. Note that the system is limited to 32 zone masks. If the system would exceed 32 zone masks, this command will return with a failure and no zone masks will be added. If a provided zone mask contains an ID matching an existing zone mask ID, the existing zone mask will be replaced by the new zone mask.

This command is available with authorization of the following access role(s):

- USER (default)

User Request Parameters

Name	Description	Type	Constraints
zone_mask_array	Contains a list of measurement zone masks	*Data_zon* []	Maximum 32 elements

Data_zone_mask

Name	Description	Type	Constraints
id	Unique identifier for the zone mask. If no ID is provided, the system will provide a unique ID.	integer	$0 \leq x \leq 31$
coordinates	Contains a series of geodetic coordinates used to define a polygon. The points must be ordered so that the defined polygon is not self-intersecting. The polygon defines the zone mask region.	*Data_lat_* []	Maximum 32 elements
agl_max	Max height above ground in meters at which to apply the zone mask. If no value is provided for agl_max, agl_max = max value.	number	$0 \leq x \leq 1000000.0$

Data_lat_long_point

Name	Description	Type	Constraints
latitude_deg	WGS84 latitude position in degrees	number	$-90.0 \leq x \leq 90.0$
longitude_deg	WGS84 longitude position in degrees	number	$-180.0 \leq x \leq 180.0$

Example Request

RadarIO Prompt Syntax

```
insert_zone_mask zone_mask_array [{id: 0, coordinates: [{latitude_deg: 0.0,␣
↪longitude_deg: 0.0}, {latitude_deg: 1.0, longitude_deg: 1.0}, {latitude_deg: 0.
↪0, longitude_deg: 2.0}], agl_max: 100.0}, {id: 1, coordinates: [{latitude_deg:␣
↪1.0, longitude_deg: 0.0}, {latitude_deg: 2.0, longitude_deg: 1.0}, {latitude_
↪deg: 1.0, longitude_deg: 2.0}], agl_max: 200.0}]
```

RadarIO Script Syntax

```
client.insert_zone_mask(zone_mask_array=[{'id': 0, 'coordinates': [{'latitude_deg
↪': 0.0, 'longitude_deg': 0.0}, {'latitude_deg': 1.0, 'longitude_deg': 1.0}, {
↪'latitude_deg': 0.0, 'longitude_deg': 2.0}], 'agl_max': 100.0}, {'id': 1,
↪'coordinates': [{'latitude_deg': 1.0, 'longitude_deg': 0.0}, {'latitude_deg':␣
↪2.0, 'longitude_deg': 1.0}, {'latitude_deg': 1.0, 'longitude_deg': 2.0}], 'agl_
↪max': 200.0}])
```

JSON Syntax (as written to control port)

```
{
  "method": "insert_zone_mask",
  "id": 86,
  "params": {
    "zone_mask_array": [
      {
        "id": 0,
        "coordinates": [
          {"latitude_deg": 0.0, "longitude_deg": 0.0},
          {"latitude_deg": 1.0, "longitude_deg": 1.0},
          {"latitude_deg": 0.0, "longitude_deg": 2.0}
        ],
        "agl_max": 100.0
```

(continues on next page)

(continued from previous page)

```
      },
      {
        "id": 1,
        "coordinates": [
          {"latitude_deg": 1.0, "longitude_deg": 0.0},
          {"latitude_deg": 2.0, "longitude_deg": 1.0},
          {"latitude_deg": 1.0, "longitude_deg": 2.0}
        ],
        "agl_max": 200.0
      }
    ]
  }
}
```

Example Response

```
{"id": 86, "result": {}}
```

license_mission

Request to add a mission license. The system must be in the service state (see *enter_service* command) to add a license.

This command is available with authorization of the following access role(s):

- USER (default)

User Request Parameters

Name	Description	Type	Constraints
mission	Configurable mission sets. NOTE: The mission set should only be updated through the configure_mission command.	enum	• C-UAS_1 • C-UAS_2 • DISMOUNT • COASTAL_C-UAS
licensekey	The license key	string	ASCII characters

Example Request

RadarIO Prompt Syntax

```
license_mission mission C-UAS_1 licensekey <The license key>
```

RadarIO Script Syntax

```
client.license_mission(mission=C-UAS_1, licensekey=<The license key>)
```

JSON Syntax (as written to control port)

```
{
  "method": "license_mission",
  "id": 84,
  "params": {"mission": "C-UAS_1", "licensekey": "<The license key>"}
}
```

Example Response

```
{"id": 84, "result": {}}
```

list_missions

Returns missions and license status.

This command is available with authorization of the following access role(s):

- USER (default)

Radar Response Parameters

Name	Description	Type	Constraints
list	Array of mission info	*mission_info* []	Maximum 32 elements

mission_info

Example Request

RadarIO Prompt Syntax

```
list_missions
```

RadarIO Script Syntax

```
client.list_missions()
```

JSON Syntax (as written to control port)

```
{"method": "list_missions", "id": 83}
```

Example Response

```
{"id": 83, "result": {"list": "<Array of mission info>"}}
```

mode_set_start

Start radar operation.

This command is available with authorization of the following access role(s):

- USER (default)

Example Request

RadarIO Prompt Syntax

```
mode_set_start
```

RadarIO Script Syntax

```
client.mode_set_start()
```

JSON Syntax (as written to control port)

```
{"method": "mode_set_start", "id": 68}
```

Example Response

```
{"id": 68, "result": {}}
```

mode_set_stop

Stop radar operation.

This command is available with authorization of the following access role(s):

- USER (default)

Example Request

RadarIO Prompt Syntax

```
mode_set_stop
```

RadarIO Script Syntax

```
client.mode_set_stop()
```

JSON Syntax (as written to control port)

```
{"method": "mode_set_stop", "id": 69}
```

Example Response

```
{"id": 69, "result": {}}
```

remove_zone_mask

Remove existing measurement based zone masks based on zone mask ID.

This command is available with authorization of the following access role(s):

- USER (default)

User Request Parameters

Name	Description	Type	Constraints
zone_mask_id_array	Defines a list of zone mask IDs	integer[]	$0 \leq x \leq 31$ Maximum 32 elements

Example Request

RadarIO Prompt Syntax

```
remove_zone_mask zone_mask_id_array [0, 1]
```

RadarIO Script Syntax

```
client.remove_zone_mask(zone_mask_id_array=[0, 1])
```

JSON Syntax (as written to control port)

```
{
  "method": "remove_zone_mask",
  "id": 87,
  "params": {"zone_mask_id_array": [0, 1]}
}
```

Example Response

```
{"id": 87, "result": {}}
```

reset_radar_params

Reset non-persistent Radar App configuration parameters (those changed by *set*) to defaults.

This command is available with authorization of the following access role(s):

- USER (default)

Example Request

RadarIO Prompt Syntax

```
reset_radar_params
```

RadarIO Script Syntax

```
client.reset_radar_params()
```

JSON Syntax (as written to control port)

```
{"method": "reset_radar_params", "id": 60}
```

Example Response

```
{"id": 60, "result": {}}
```

set

Set one or more RadarApp configuration parameters. The system must be in the service state if one or more provided RadarApp configuration parameter's defined *Modifiable Mode* does not include *Operation* (see *enter_service* command) to set.

This command is available with authorization of the following access role(s):

- USER (default)

User Request Parameters

List of RadarApp parameter, value pairs. See the *RadarApp Parameters* section.

Example Request

RadarIO Prompt Syntax

```
set search_azmin -50 search_azmax 50
```

RadarIO Script Syntax

```
client.set(search_azmin=-50, search_azmax=50)
```

JSON Syntax (as written to control port)

```
{
  "method": "set", "id": 61, "params": {"search_azmin": -50, "search_azmax": 50}
}
```

Example Response

```
{"id": 61, "result": {}}
```

set_radar_kinematics

Update the radar's position and orientation in the world. This is an alternative to updating the reference frame, and ignores the reference frame and radar-to-reference kinematics. The radar kinematics will only be reset upon a power cycle.

This command is available with authorization of the following access role(s):

- USER (default)

User Request Parameters

Name	Description	Type	Constraints
latitude_deg	WGS84 latitude of the radar frame origin in degrees	number	$-90.0 \leq x \leq 90.0$
longitude_deg	WGS84 longitude of the radar frame origin in degrees	number	$-180.0 \leq x \leq 180.0$
altitude_m	Altitude of the radar frame origin in meters. The vertical datum used to define the altitude can be specified with the altitude_type field. When defining the ellipsoidal altitude, the altitude should be referenced from the WGS84 ellipsoid. If defining the orthometric altitude, the altitude should be referenced from the EGM96 geoid.	number	$-1000.0 \leq x \leq 1000000.0$
heading_deg	Number of degrees clockwise from north that the radar's z-axis (broadside) is pointed	number	$-360.0 \leq x \leq 360.0$
pitchback_deg	Number of degrees above the horizon (the radar's level plane) that radar z-axis (broadside) is pointing	number	$-360.0 \leq x \leq 360.0$
tilt_deg	The left-handed rotation degrees around radar z-axis (broadside)	number	$-360.0 \leq x \leq 360.0$

continues on next page

Table 50 – continued from previous page

Name	Description	Type	Constraints
altitude_type	Defines the vertical datum used for the given altitude. The ellipsoidal height/altitude refers to the height above the ellipsoid. Echoshield assumes use of the WGS84 ellipsoid. Orthometric altitude is the height above the geoid, which represents mean sea level and is determined by Earth's gravity. Echoshield assumes use of the EGM96 geoid, but should not be sensitive to other geoid models. It is important to explicitly define whether a given altitude is the height above the ellipsoid or geoid as these can differ by as much as ~90 meters depending on where you are in the world.	enum	• ellipsoidal • orthometric
unix_time_ns	**Optional** Optional: the Unix timestamp of this kinematics packet, expressed in nanoseconds	integer	$x \geq 0$

Radar Response Parameters

Name	Description	Type	Constraints
kinematics_id	Reference kinematics update ID.	integer	'uint64_t' range
kinematics_sensor_agreement	How closely the user-set kinematics agrees with the internal INS solution.	string	ASCII characters

Example Request

RadarIO Prompt Syntax

```
set_radar_kinematics latitude_deg 47.7088 longitude_deg -122.1878 altitude_m 30␣
↪heading_deg 0 pitchback_deg 0 tilt_deg 0 altitude_type orthometric unix_time_
↪ns 1734387877000000000
```

RadarIO Script Syntax

```
client.set_radar_kinematics(latitude_deg=47.7088, longitude_deg=-122.1878,␣
↪altitude_m=30, heading_deg=0, pitchback_deg=0, tilt_deg=0, altitude_
↪type=orthometric, unix_time_ns=1734387877000000000)
```

JSON Syntax (as written to control port)

```
{
  "method": "set_radar_kinematics",
  "id": 70,
  "params": {
    "latitude_deg": 47.7088,
    "longitude_deg": -122.1878,
    "altitude_m": 30,
    "heading_deg": 0,
    "pitchback_deg": 0,
    "tilt_deg": 0,
    "altitude_type": "orthometric",
    "unix_time_ns": 1734387877000000000
  }
}
```

Example Response

```
{
  "id": 70,
  "result": {
    "kinematics_id": 0,
    "kinematics_sensor_agreement": "User-set kinematics agrees with internal␣
↪sensors."
  }
}
```

set_reference_frame_offsets

Set the offsets of the radar frame with respect to the reference frame, which is required before setting reference kinematics. When updating the reference kinematics with set_reference_kinematics_ecef and _geodetic, radar kinematics will automatically update using these offsets. The rotational offsets are Tait-Bryan Euler angles applied intrinsically in the Yaw, Pitch, Roll order.

This command is available with authorization of the following access role(s):

- USER (default)

User Request Parameters

Name	Description	Type	Constraints
position_relative_to _reference_x	The position of the radar frame on the reference-frame's x-axis, in meters	number	$-1000000 \leq x \leq 1000000$
position_relative_to _reference_y	The position of the radar frame on the reference-frame's y-axis, in meters	number	$-1000000 \leq x \leq 1000000$
position_relative_to _reference_z	The position of the radar frame on the reference-frame's z-axis, in meters	number	$-1000000 \leq x \leq 1000000$
rotation_relative_to _reference_yaw	The right-handed yaw of the radar frame around the reference-frame's z-axis, in degrees. This is the first rotation applied in the Yaw-Pitch-Roll sequence, meaning the rotation is applied relative to the original reference-frame.	number	$-360.0 \leq x \leq 360.0$
rotation_relative_to _reference_pitch	The right-handed pitch of the radar frame around the yaw-rotated frame's y-axis, in degrees. This is the second rotation in the Yaw-Pitch-Roll sequence, meaning it is applied after the yaw rotation and is relative to the yaw-rotated frame.	number	$-360.0 \leq x \leq 360.0$

continues on next page

Table 52 – continued from previous page

Name	Description	Type	Constraints
rotation_relative_to _reference_roll	The right-handed roll of the radar frame around the yaw-and-pitch-rotated frame's x-axis, in degrees. This is the final rotation in the Yaw-Pitch-Roll sequence, meaning it is applied after both yaw and pitch rotations, relative to the already-rotated frame.	number	$-360.0 \leq x \leq 360.0$

Example Request

RadarIO Prompt Syntax

```
set_reference_frame_offsets position_relative_to_reference_x -460827.77 position_
↪relative_to_reference_y -276177.3 position_relative_to_reference_z -729158.39␣
↪rotation_relative_to_reference_yaw 90.05 rotation_relative_to_reference_pitch -
↪302.51 rotation_relative_to_reference_roll 84.53
```

RadarIO Script Syntax

```
client.set_reference_frame_offsets(position_relative_to_reference_x=-460827.77,␣
↪position_relative_to_reference_y=-276177.3, position_relative_to_reference_z=-
↪729158.39, rotation_relative_to_reference_yaw=90.05, rotation_relative_to_
↪reference_pitch=-302.51, rotation_relative_to_reference_roll=84.53)
```

JSON Syntax (as written to control port)

```
{
  "method": "set_reference_frame_offsets",
  "id": 71,
  "params": {
    "position_relative_to_reference_x": -460827.77,
    "position_relative_to_reference_y": -276177.3,
    "position_relative_to_reference_z": -729158.39,
    "rotation_relative_to_reference_yaw": 90.05,
    "rotation_relative_to_reference_pitch": -302.51,
    "rotation_relative_to_reference_roll": 84.53
  }
}
```

Example Response


```
{"id": 71, "result": {}}
```

set_reference_kinematics_ecef

Update reference frame kinematics values. This, combined with the reference frame offsets, defines the position of the radar in the world. The set_reference_frame_offsets command is required to be sent before this command can be used. The state of the reference frame will only be reset upon a power cycle. This is an alternative to set_radar_kinematics. See the *Reference Frame Guide* section for a detailed description of kinematics.

This command is available with authorization of the following access role(s):

- USER (default)

User Request Parameters

Name	Description	Type	Constraints
position_ecef_x	The x component of the Earth Centered Earth Fixed position of the reference frame in meters	number	$-100000000.0 \leq x \leq 100000000.0$
position_ecef_y	The y component of the Earth Centered Earth Fixed position of the reference frame in meters	number	$-100000000.0 \leq x \leq 100000000.0$
position_ecef_z	The z component of the Earth Centered Earth Fixed position of the reference frame in meters	number	$-100000000.0 \leq x \leq 100000000.0$
quaternion_ecef_w	The w (real) component of the w+x*i+y*j+z*k quaternion of the reference frame. This quaternion defines the rotation of the reference frame relative to the Earth Centered Earth Fixed frame and should have unit magnitude.	number	$-1 \leq x \leq 1$

continues on next page

Table 53 – continued from previous page

Name	Description	Type	Constraints
quaternion_ecef_x	The x (real) component of the w+x*i+y*j+z*k quaternion of the reference frame. This quaternion defines the rotation of the reference frame relative to the Earth Centered Earth Fixed frame and should have unit magnitude.	number	$-1 \leq x \leq 1$
quaternion_ecef_y	The y (real) component of the w+x*i+y*j+z*k quaternion of the reference frame. This quaternion defines the rotation of the reference frame relative to the Earth Centered Earth Fixed frame and should have unit magnitude.	number	$-1 \leq x \leq 1$
quaternion_ecef_z	The z (real) component of the w+x*i+y*j+z*k quaternion of the reference frame. This quaternion defines the rotation of the reference frame relative to the Earth Centered Earth Fixed frame and should have unit magnitude.	number	$-1 \leq x \leq 1$
velocity_ecef_x	The x component of the Earth Centered Earth Fixed linear velocity of the reference frame, in meters per second	number	$-1000000 \leq x \leq 1000000$
velocity_ecef_y	The y component of the Earth Centered Earth Fixed linear velocity of the reference frame, in meters per second	number	$-1000000 \leq x \leq 1000000$
velocity_ecef_z	The z component of the Earth Centered Earth Fixed linear velocity of the reference frame, in meters per second	number	$-1000000 \leq x \leq 1000000$
rotation_rate_ecef_x	The x component of the Earth Centered Earth Fixed rotational velocity vector of the reference frame, in radians per second	number	$-1000000 \leq x \leq 1000000$

continues on next page

Table 53 – continued from previous page

Name	Description	Type	Constraints
rotation_rate_ecef_y	The y component of the Earth Centered Earth Fixed rotational velocity vector of the reference frame, in radians per second	number	$-1000000 \leq x \leq 1000000$
rotation_rate_ecef_z	The z component of the Earth Centered Earth Fixed rotational velocity vector of the reference frame, in radians per second	number	$-1000000 \leq x \leq 1000000$
unix_time_ns	**Optional** Optional: the Unix timestamp of this kinematics packet, expressed in nanoseconds	integer	$x \geq 0$

Radar Response Parameters

Name	Description	Type	Constraints
kinematics_id	Reference kinematics update ID.	integer	'uint64_t' range
kinematics_sensor_agreement	How closely the user-set kinematics agrees with the internal INS solution.	string	ASCII characters

Example Request

RadarIO Prompt Syntax

```
set_reference_kinematics_ecef position_ecef_x -2290457.46 position_ecef_y -
↪3638871 position_ecef_z 4695174.693 quaternion_ecef_w 0.1744 quaternion_ecef_x␣
↪-0.8165 quaternion_ecef_y -0.4508 quaternion_ecef_z -0.3158 velocity_ecef_x 0␣
↪velocity_ecef_y 0 velocity_ecef_z 0 rotation_rate_ecef_x 0 rotation_rate_ecef_
↪y 0 rotation_rate_ecef_z 0 unix_time_ns 1734387877000000000
```

RadarIO Script Syntax

```
client.set_reference_kinematics_ecef(position_ecef_x=-2290457.46, position_ecef_
```

(continues on next page)

(continued from previous page)

```
↪y=-3638871, position_ecef_z=4695174.693, quaternion_ecef_w=0.1744, quaternion_
↪ecef_x=-0.8165, quaternion_ecef_y=-0.4508, quaternion_ecef_z=-0.3158, velocity_
↪ecef_x=0, velocity_ecef_y=0, velocity_ecef_z=0, rotation_rate_ecef_x=0,␣
↪rotation_rate_ecef_y=0, rotation_rate_ecef_z=0, unix_time_
↪ns=1734387877000000000)
```

JSON Syntax (as written to control port)

```
{
  "method": "set_reference_kinematics_ecef",
  "id": 73,
  "params": {
    "position_ecef_x": -2290457.46,
    "position_ecef_y": -3638871,
    "position_ecef_z": 4695174.693,
    "quaternion_ecef_w": 0.1744,
    "quaternion_ecef_x": -0.8165,
    "quaternion_ecef_y": -0.4508,
    "quaternion_ecef_z": -0.3158,
    "velocity_ecef_x": 0,
    "velocity_ecef_y": 0,
    "velocity_ecef_z": 0,
    "rotation_rate_ecef_x": 0,
    "rotation_rate_ecef_y": 0,
    "rotation_rate_ecef_z": 0,
    "unix_time_ns": 1734387877000000000
  }
}
```

Example Response

```
{
  "id": 73,
  "result": {
    "kinematics_id": 0,
    "kinematics_sensor_agreement": "User-set kinematics agrees with internal␣
↪sensors."
  }
}
```

set_reference_kinematics_geodetic

Update reference frame's position and orientation in the world. This, combined with the reference frame offsets, defines the position of the radar in the world. The set_reference_frame_offsets command is required to be sent before this command can be used. Rotations are intrinsic Tait-Bryan Euler Angles in Yaw, Pitch, Roll order and relative to the local position's ENU frame. The state of the reference frame will only be reset upon a power cycle. See the *Reference Frame Guide* section for a detailed description of kinematics.

This command is available with authorization of the following access role(s):

- USER (default)

User Request Parameters

Name	Description	Type	Constraints
latitude_deg	WGS84 latitude of the reference frame origin in degrees	number	$-90.0 \leq x \leq 90.0$
longitude_deg	WGS84 longitude of the reference frame origin in degrees	number	$-180.0 \leq x \leq 180.0$
altitude_m	Altitude of the reference frame origin in meters. The vertical datum used to define the altitude can be specified with the altitude_type field. By default, the altitude is assumed to be the orthometric height (MSL). If defining the ellipsoidal altitude, the altitude should be referenced from the WGS84 ellipsoid. If defining the orthometric altitude, the altitude should be referenced from the EGM96 geoid.	number	$-1000.0 \leq x \leq 1000000.0$
yaw_enu_deg	The right-handed yaw of the radar frame around the ENU frame's z-axis, in degrees. This is the first rotation applied in the Yaw-Pitch-Roll sequence, meaning the rotation is applied relative to the ENU frame	number	$-360.0 \leq x \leq 360.0$

continues on next page

Table 55 – continued from previous page

Name	Description	Type	Constraints
pitch_enu_deg	The right-handed pitch of the radar frame around the yaw-rotated frame's y-axis, in degrees. This is the second rotation in the Yaw-Pitch-Roll sequence, meaning it is applied after the yaw rotation and is relative to the yaw-rotated frame.	number	$-360.0 \leq x \leq 360.0$
roll_enu_deg	The right-handed roll of the radar frame around the pitch-and-roll-rotated frame's x-axis, in degrees. This is the final rotation in the Yaw-Pitch-Roll sequence, meaning it is applied after both yaw and pitch rotations, relative to the already-rotated frame.	number	$-360.0 \leq x \leq 360.0$
altitude_type	Defines the vertical datum used for the given altitude. The ellipsoidal height/altitude refers to the height above the ellipsoid. Echoshield assumes use of the WGS84 ellipsoid. Orthometric altitude is the height above the geoid, which represents mean sea level and is determined by Earth's gravity. Echoshield assumes use of the EGM96 geoid, but should not be sensitive to other geoid models. It is important to explicitly define whether a given altitude is the height above the ellipsoid or geoid as these can differ by as much as ~90 meters depending on where you are in the world.	enum	• ellipsoidal • orthometric

continues on next page

Table 55 – continued from previous page

Name	Description	Type	Constraints
unix_time_ns	**Optional** Optional: the Unix timestamp of this kinematics packet, expressed in nanoseconds	integer	$x \geq 0$

Radar Response Parameters

Name	Description	Type	Constraints
kinematics_id	Reference kinematics update ID.	integer	'uint64_t' range
kinematics_sensor_agreement	How closely the user-set kinematics agrees with the internal INS solution.	string	ASCII characters

Example Request

RadarIO Prompt Syntax

```
set_reference_kinematics_geodetic latitude_deg 47.7088 longitude_deg -122.1878␣
↪altitude_m 30 yaw_enu_deg 90 pitch_enu_deg -110 roll_enu_deg 0 altitude_type␣
↪orthometric unix_time_ns 1734387877000000000
```

RadarIO Script Syntax

```
client.set_reference_kinematics_geodetic(latitude_deg=47.7088, longitude_deg=-
↪122.1878, altitude_m=30, yaw_enu_deg=90, pitch_enu_deg=-110, roll_enu_deg=0,␣
↪altitude_type=orthometric, unix_time_ns=1734387877000000000)
```

JSON Syntax (as written to control port)

```
{
  "method": "set_reference_kinematics_geodetic",
  "id": 74,
  "params": {
    "latitude_deg": 47.7088,
    "longitude_deg": -122.1878,
    "altitude_m": 30,
```

(continues on next page)

(continued from previous page)

```
    "yaw_enu_deg": 90,
    "pitch_enu_deg": -110,
    "roll_enu_deg": 0,
    "altitude_type": "orthometric",
    "unix_time_ns": 1734387877000000000
  }
}
```

Example Response

```
{
  "id": 74,
  "result": {
    "kinematics_id": 0,
    "kinematics_sensor_agreement": "User-set kinematics agrees with internal␣
↪sensors."
  }
}
```

`spectrum_test_start`

Starts a spectrum test using a beam defined by the channel, bandwidth and pulse length parameters.

This command is available with authorization of the following access role(s):

- SUPER_USER

User Request Parameters

Name	Description	Type	Constraints
tx_channel	Radar transmit channel	enum	• L1 • L2 • L3 • L4 • L5 • U1 • U2 • U3 • U4 • U5 • U6 • U7 • U8 • U9 • U10 • U11 • U12 • U13 • U14 • U15 • U16 • U17
bandwidth	Transmit bandwidth	enum	• BANDWIDTH_0MHZ • BANDWIDTH_25MHZ • BANDWIDTH_50MHZ • BANDWIDTH_100MHZ • BANDWIDTH_200MHZ

continues on next page

Table 57 – continued from previous page

Name	Description	Type	Constraints
pulse_length	Transmit pulse length	enum	• PULSE_LENGTH_MIN • PULSE_LENGTH_MAX

Example Request

RadarIO Prompt Syntax

```
spectrum_test_start tx_channel U5 bandwidth BANDWIDTH_0MHZ pulse_length PULSE_
↪LENGTH_MAX
```

RadarIO Script Syntax

```
client.spectrum_test_start(tx_channel=U5, bandwidth=BANDWIDTH_0MHZ, pulse_
↪length=PULSE_LENGTH_MAX)
```

JSON Syntax (as written to control port)

```
{
  "method": "spectrum_test_start",
  "id": 89,
  "params": {
    "tx_channel": "U5",
    "bandwidth": "BANDWIDTH_0MHZ",
    "pulse_length": "PULSE_LENGTH_MAX"
  }
}
```

Example Response

```
{"id": 89, "result": {}}
```

spectrum_test_stop

Stops a spectrum test.

This command is available with authorization of the following access role(s):

- SUPER_USER

Example Request

RadarIO Prompt Syntax

```
spectrum_test_stop
```

RadarIO Script Syntax

```
client.spectrum_test_stop()
```

JSON Syntax (as written to control port)

```
{"method": "spectrum_test_stop", "id": 90}
```

Example Response

```
{"id": 90, "result": {}}
```

track_focus_start

Start focusing on a particular track. This command requires the user to give a track identifier (uuid or trackId), the particular type of track identifier (TRACK_UUID, HANDOFF_UUID, or TRACK_ID), and the type of track focus desired (PRIORITY_TRACK or SINGLE_TARGET_TRACK). 'uuid' and 'track_id' are semi-optional (only one is required), while 'track_identifier_type' and 'focus_type' are both required. If identifying the track via TRACK_UUID or HANDOFF_UUID, the 'uuid' field must be given. If identifying the track with TRACK_ID, the 'trackId' field must be given. If an optional field is given that does not correspond with 'track_identifier_type', it is ignored. If selecting the PRIORITY_TRACK focus type, the system will reallocate all of its search and classification beam resources to interrogate the region around the specified track, but will continue to look back at existing tracks. If selecting the SINGLE_TARGET_TRACK focus type, the system will reallocate all of its beam resources to interrogate the specified track. This will result in all other tracks not in the vicinity of the selected target to coast until they are eventually killed. In both cases, the radar will continue to acquire new targets in the vicinity of the selected target. PRIORITY_TRACK should be used in cases where the user wants to maintain existing tracks while SINGLE_TARGET_TRACK should be used if the user wants to maximize accuracy on the specified track.

This command is available with authorization of the following access role(s):

- USER (default)

User Request Parameters

Name	Description	Type	Constraints
uuid	**Optional** Byte array representation of UUID	integer[16]	$0 \leq x \leq 255$
track_id	**Optional** Unique incrementing identifier for tracks (more human readable than a UUID)	integer	'uint64_t' range
track_identifier_type	Track identifier type	enum	• TRACK_UUID • HANDOFF_UUID • TRACK_ID
focus_type	Track focus type	enum	• PRIORITY_TRACK • SINGLE_TARGET_TRACK

Radar Response Parameters

Name	Description	Type	Constraints
track_found	Whether or not the requested track was found	boolean	[True, False]

Example Request

RadarIO Prompt Syntax

```
track_focus_start uuid [112, 228, 171, 68, 209, 39, 195, 235, 235, 195, 198, 79,␣
↪186, 98, 53, 199] track_id 14016529498783923983 track_identifier_type TRACK_ID␣
↪focus_type SINGLE_TARGET_TRACK
```

RadarIO Script Syntax

```
client.track_focus_start(uuid=[112, 228, 171, 68, 209, 39, 195, 235, 235, 195,␣
↪198, 79, 186, 98, 53, 199], track_id=14016529498783923983, track_identifier_
↪type=TRACK_ID, focus_type=SINGLE_TARGET_TRACK)
```

JSON Syntax (as written to control port)

```
{
  "method": "track_focus_start",
  "id": 65,
  "params": {
    "uuid": [
      112, 228, 171, 68, 209, 39, 195, 235, 235, 195, 198, 79, 186, 98, 53, 199
    ],
    "track_id": 14016529498783923983,
    "track_identifier_type": "TRACK_ID",
    "focus_type": "SINGLE_TARGET_TRACK"
  }
}
```

Example Response

```
{
  "id": 65,
  "result": {"track_found": "<Whether or not the requested track was found>"}
}
```

track_focus_stop

Stop focusing on any particular track.

This command is available with authorization of the following access role(s):

- USER (default)

Example Request

RadarIO Prompt Syntax

```
track_focus_stop
```

RadarIO Script Syntax

```
client.track_focus_stop()
```

JSON Syntax (as written to control port)

```
{"method": "track_focus_stop", "id": 66}
```

Example Response

```
{"id": 66, "result": {}}
```

track_handoff

Handoff track estimate to the radar that was generated from another source. The user must provide the optional estimate parameter that matches the format parameter. The time the estimate was measured must also be provided in the unix_time_ns parameter and the time sources must be synchronized with the radar. The uuid parameter will populate the track's handoff_uuid field. If uuid is not provided one will be generated on the radar and sent in the command response so the user can associate the data. The rcs_dBsm and rcs_std_dBsm optional parameters will only be used if both are provided.

> Track handoffs should be limited to 1000 per second when RCS values are provided and to 2 per second when no RCS values are provided. Exceeding these handoff limits can cause a reduction in the track update rate, degradation in track accuracy and degradation in track acquisition performance.

This command is available with authorization of the following access role(s):

- USER (default)

User Request Parameters

Name	Description	Type	Constraints
format	Enumerates which handoff format is expected in a track_handoff command	enum	• RADAR_SPHERICAL • GEODETIC • RADAR_ENU_POSITION • RADAR_ENU_FULL • ECEF_POSITION • ECEF_FULL
unix_time_ns	The Unix time of track estimate, expressed in nanoseconds. First the user should make sure that the radar time source is synchronized.	integer	$x \geq 0$
uuid	**Optional** Byte array representation of UUID	integer[16]	$0 \leq x \leq 255$
rcs_dBsm	**Optional** Defines target RCS estimate in dBsm, only used if rcs_std_dBsm is also provided	number	$-100 \leq x \leq 100$

continues on next page

Table 60 – continued from previous page

Name	Description	Type	Constraints
rcs_std_dBsm	**Optional** Defines target RCS standard deviation in dBsm, only used if rcs_dBsm is also provided	number	$0 < x \leq 100$
radar_spherical	**Optional** Defines track estimate in radar relative spherical coordinates	*radar_spi*	
geodetic	**Optional** Defines track estimate in geodetic coordinates	*geode-tic*	
radar_enu_position	**Optional** Defines track position estimate in radar relative ENU coordinates.	*radar_en*	
radar_enu_full	**Optional** Defines track position and velocity estimate in radar relative ENU coordinates.	*radar_en*	
ecef_position	**Optional** Defines track position estimate and covariance in ECEF coordinates	*ecef_posi*	
ecef_full	**Optional** Defines track estimate in full ECEF coordinates with complete covariance matrix.	*ecef_full*	

radar_spherical

Name	Description	Type	Constraints
az_deg	Track radar relative azimuth estimate in degrees	number	$-90 \leq x \leq 90$
el_deg	Track radar relative elevation estimate in degrees	number	$-90 \leq x \leq 90$
range_m	Track radar relative range in meters	number	$0 < x \leq 25000$
az_std_deg	Track azimuth estimate standard deviation in degrees	number	$x > 0$

continues on next page

Table 61 – continued from previous page

Name	Description	Type	Constraints
el_std_deg	Track elevation estimate standard deviation in degrees	number	$x > 0$
range_std_m	Track range estimate standard deviation in meters	number	$x > 0$

geodetic

Name	Description	Type	Constraints
latitude_deg	Track latitude estimate in degrees	number	$-90 \leq x \leq 90$
longitude_deg	Track longitude estimate in degrees	number	$-180 \leq x \leq 180$
altitude_m	Track altitude with respect to defined datum in meters	number	$-1000.0 \leq x \leq 1000000$
altitude_type	Defines the vertical datum used for the given altitude. The ellipsoidal height/altitude refers to the height above the ellipsoid. Echoshield assumes use of the WGS84 ellipsoid. Orthometric altitude is the height above the geoid, which represents mean sea level and is determined by Earth's gravity. Echoshield assumes use of the EGM96 geoid, but should not be sensitive to other geoid models. It is important to explicitly define whether a given altitude is the height above the ellipsoid or geoid as these can differ by as much as ~90 meters depending on where you are in the world.	enum	• ellipsoidal • orthometric

continues on next page

Table 62 – continued from previous page

Name	Description	Type	Constraints
horizontal_std_m	Track handoff estimate standard deviation in the horizontal plane (local east, north plane) in meters	number	$x > 0$
vertical_std_m	Track handoff estimate standard deviation in the local up direction in meters	number	$x > 0$

radar_enu_position

Name	Description	Type	Constraints
east_pos_m	Track radar relative east position in meters.	number	$-25000 \leq x \leq 25000$
north_pos_m	Track radar relative north position in meters.	number	$-25000 \leq x \leq 25000$
up_pos_m	Track radar relative up position in meters.	number	$-25000 \leq x \leq 25000$
east_pos_std_m	Track east estimate standard deviation in meters.	number	$x > 0$
north_pos_std_m	Track north estimate standard deviation in meters.	number	$x > 0$
up_pos_std_m	Track up estimate standard deviation in meters.	number	$x > 0$

radar_enu_full

Name	Description	Type	Constraints
east_pos_m	Track radar relative east position in meters	number	$-25000 \leq x \leq 25000$
north_pos_m	Track radar relative north position in meters	number	$-25000 \leq x \leq 25000$
up_pos_m	Track radar relative up position in meters	number	$-25000 \leq x \leq 25000$
east_pos_std_m	Track east estimate standard deviation in meters	number	$x > 0$

continues on next page

Table 64 – continued from previous page

Name	Description	Type	Constraints
north_pos_std_m	Track north estimate standard deviation in meters	number	$x > 0$
up_pos_std_m	Track up estimate standard deviation in meters	number	$x > 0$
east_vel_m_s	Track radar relative east velocity in meters per second	number	$-10000 \leq x \leq 10000$
north_vel_m_s	Track radar relative north velocity in meters per second	number	$-10000 \leq x \leq 10000$
up_vel_m_s	Track radar relative up velocity in meters per second	number	$-10000 \leq x \leq 10000$
east_vel_std_m_s	Track east velocity estimate standard deviation in meters per second	number	$x > 0$
north_vel_std_m_s	Track north velocity estimate standard deviation in meters per second	number	$x > 0$
up_vel_std_m_s	Track up velocity estimate standard deviation in meters per second	number	$x > 0$

ecef_position

Name	Description	Type	Constraints
x_m	Track ECEF x position in meters	number	'double' range
y_m	Track ECEF y position in meters	number	'double' range
z_m	Track ECEF z position in meters	number	'double' range
ecef_pos_covariance	A float[6] array corresponding to the upper triangular representation of ecef position covariance. Elements are ordered as [xx, xy, xz, yy, yz, zz].	number[6]	'float' range

ecef_full

Name	Description	Type	Constraints
ecef_pos_est	A double[3] array representing the track's ECEF position. This matches the format used in the gen3 track packet.	number[3]	'double' range
ecef_vel_est	A float[3] array representing the track's ECEF velocity. This matches the format used in the gen3 track packet.	number[3]	'float' range
ecef_state_covarianc e	A float[21] array corresponding to the upper triangular representation of the track's ECEF estimate covariance. This matches the format used in the gen3 track packet.	number[21]	'float' range

Radar Response Parameters

Name	Description	Type	Constraints
handoff_uuid	Byte array representation of UUID	integer[16]	$0 \leq x \leq 255$

Example Request

RadarIO Prompt Syntax

```
track_handoff format RADAR_SPHERICAL unix_time_ns 1748540293823925314 uuid [218,␣
↪48, 18, 238, 100, 21, 189, 87, 205, 240, 150, 165, 14, 9, 200, 127] rcs_dBsm -
↪20.0 rcs_std_dBsm 3.0 radar_spherical {az_deg: 12.5, el_deg: 0.0, range_m:␣
↪2000, az_std_deg: 1, el_std_deg: 1, range_std_m: 10}
```

RadarIO Script Syntax

```
client.track_handoff(format=RADAR_SPHERICAL, unix_time_ns=1748540293823925314,␣
↪uuid=[218, 48, 18, 238, 100, 21, 189, 87, 205, 240, 150, 165, 14, 9, 200, 127],
↪ rcs_dBsm=-20.0, rcs_std_dBsm=3.0, radar_spherical={'az_deg': 12.5, 'el_deg':␣
↪0.0, 'range_m': 2000, 'az_std_deg': 1, 'el_std_deg': 1, 'range_std_m': 10})
```

JSON Syntax (as written to control port)


```
{
  "method": "track_handoff",
  "id": 67,
  "params": {
    "format": "RADAR_SPHERICAL",
    "unix_time_ns": 1748540293823925314,
    "uuid": [
      218, 48, 18, 238, 100, 21, 189, 87, 205, 240, 150, 165, 14, 9, 200, 127
    ],
    "rcs_dBsm": -20.0,
    "rcs_std_dBsm": 3.0,
    "radar_spherical": {
      "az_deg": 12.5,
      "el_deg": 0.0,
      "range_m": 2000,
      "az_std_deg": 1,
      "el_std_deg": 1,
      "range_std_m": 10
    }
  }
}
```


Example Response


```
{"id": 67, "result": {}}
```

3.6 Field Upgrade Commands

Commands that implement the Field Upgrade process.

NOTE: None of the commands listed in this section should be used directly. Instead, use the RadarIO EchoShield Field Upgrader:

```
$ radario echoshield upgrade <command> [opts]
```

See the RadarIO User Manual for more information.

- *apply_staged_field_upgrade*
- *list_staged_field_upgrades*
- *remove_staged_field_upgrade*
- *revert_field_upgrade*
- *stage_field_upgrade*

apply_staged_field_upgrade

Apply a staged field upgrade image and reboot (will lose connection after response)

This command is available with authorization of the following access role(s):

- FIELD_UPGRADE_USER

Example Request

RadarIO Prompt Syntax

```
apply_staged_field_upgrade
```

RadarIO Script Syntax

```
client.apply_staged_field_upgrade()
```

JSON Syntax (as written to control port)

```
{"method": "apply_staged_field_upgrade", "id": 35}
```

Example Response

```
{"id": 35, "result": {}}
```

list_staged_field_upgrades

Get running and staged field upgrade images

This command is available with authorization of the following access role(s):

- FIELD_UPGRADE_USER

Radar Response Parameters

Name	Description	Type	Constraints
list	Upgrade package names	string[]	ASCII characters Maximum 2 elements

Example Request

RadarIO Prompt Syntax

```
list_staged_field_upgrades
```

RadarIO Script Syntax

```
client.list_staged_field_upgrades()
```

JSON Syntax (as written to control port)

```
{"method": "list_staged_field_upgrades", "id": 36}
```

Example Response

```
{"id": 36, "result": {"list": "<Upgrade package names>"}}
```

remove_staged_field_upgrade

Remove the staged upgrade image

This command is available with authorization of the following access role(s):

- FIELD_UPGRADE_USER

Example Request

RadarIO Prompt Syntax

```
remove_staged_field_upgrade
```

RadarIO Script Syntax

```
client.remove_staged_field_upgrade()
```

JSON Syntax (as written to control port)

```
{"method": "remove_staged_field_upgrade", "id": 37}
```

Example Response

```
{"id": 37, "result": {}}
```

revert_field_upgrade

Revert the field upgrade to the previous image if it is still there (will lose connection after response)

This command is available with authorization of the following access role(s):

- FIELD_UPGRADE_USER

Example Request

RadarIO Prompt Syntax

```
revert_field_upgrade
```

RadarIO Script Syntax

```
client.revert_field_upgrade()
```

JSON Syntax (as written to control port)

```
{"method": "revert_field_upgrade", "id": 38}
```

Example Response

```
{"id": 38, "result": {}}
```

stage_field_upgrade

Stage one chunk of the field upgrade file being staged

This command is available with authorization of the following access role(s):

- FIELD_UPGRADE_USER

User Request Parameters

Name	Description	Type	Constraints
filename	File name of the field upgrade image to stage	string	ASCII characters
i_chunk	the index of this chunk in all chunks, 0 .. n_chunk - 1	integer	'uint32_t' range
n_chunk	the total number of chunks to expect, total_image_bytes // 2MB + 1	integer	'uint32_t' range
n_byte_total	the number of total bytes in the upgrade image	integer	'uint32_t' range
md5	**Optional** md5 of the unencrypted image. Sent on first chunk only.	string	ASCII characters
b64_data	the base 64 encoded data payload for this chunk	string	ASCII characters

Example Request

RadarIO Prompt Syntax

```
stage_field_upgrade filename <File name of the field upgrade image to stage> i_
↪chunk 2589218560 n_chunk 3631690693 n_byte_total 3818052292 md5 <md5 of the␣
↪unencrypted image. Sent on first chunk only.> b64_data <the base 64 encoded␣
↪data payload for this chunk>
```

RadarIO Script Syntax

```
client.stage_field_upgrade(filename=<File name of the field upgrade image to␣
↪stage>, i_chunk=2589218560, n_chunk=3631690693, n_byte_total=3818052292, md5=
↪<md5 of the unencrypted image. Sent on first chunk only.>, b64_data=<the base␣
↪64 encoded data payload for this chunk>)
```

JSON Syntax (as written to control port)

```
{
  "method": "stage_field_upgrade",
  "id": 34,
  "params": {
    "filename": "<File name of the field upgrade image to stage>",
    "i_chunk": 2589218560,
    "n_chunk": 3631690693,
    "n_byte_total": 3818052292,
    "md5": "<md5 of the unencrypted image. Sent on first chunk only.>",
    "b64_data": "<the base 64 encoded data payload for this chunk>"
  }
}
```

Example Response

```
{"id": 34, "result": {}}
```

3.7 Device Commands

Commands facilitating access to internal devices

- *get_gps_data*
- *get_imu_data*

get_gps_data

Get GPS data. GPS mode will be reported as either 2D_Fix or 3D_Fix, assuming there are sufficient satellites in view. Note: get_gps_data will always report altitude as orthometric height above the geoid to match what is reported by the internal sensor. If ellipsoidal height is needed, the get_ins_data can be used as long as the get_gps_data command is returning a GPS mode of 3D_Fix, otherwise the reported altitude may not be correct.

This command is available with authorization of the following access role(s):

- SUPER_USER

Radar Response Parameters

Name	Description	Type	Constraints
mode	Mode of GPS fix	string	ASCII characters
time	Time of update	string	ASCII characters
latitude	Latitude (Decimal degrees, S < 0 < N)	number	$-90.0 \leq x \leq 90.0$
epy	Latitude position uncertainty, meters	number	'double' range
longitude	Longitude (Decimal degrees, W < 0 < E)	number	$-180.0 \leq x \leq 180.0$
epx	Longitude position uncertainty, meters	number	'double' range
altitude	Orthometric Altitude (MSL), meters	number	'double' range
epv	Vertical position uncertainty, meters	number	'double' range
speed	Speed (m/s)	number	'double' range
eps	Speed uncertainty, meters/sec	number	'double' range
heading	True compass heading	number	$0.0 \leq x \leq 360.0$
satsInView	Number of satellites in view	integer	'int' range
satsUsed	Number of satellites in use	integer	'int' range

Example Request

RadarIO Prompt Syntax

```
get_gps_data
```

RadarIO Script Syntax

```
client.get_gps_data()
```

JSON Syntax (as written to control port)

```
{"method": "get_gps_data", "id": 18}
```

Example Response

```
{
  "id": 18,
  "result": {
    "mode": "<Mode of GPS fix>",
    "time": "<Time of update>",
    "latitude": "<Latitude (Decimal degrees, S < 0 < N)>",
    "epy": "<Latitude position uncertainty, meters>",
    "longitude": "<Longitude (Decimal degrees, W < 0 < E)>",
    "epx": "<Longitude position uncertainty, meters>",
    "altitude": "<Orthometric Altitude (MSL), meters>",
    "epv": "<Vertical position uncertainty, meters>",
    "speed": "<Speed (m/s)>",
    "eps": "<Speed uncertainty, meters/sec>",
    "heading": "<True compass heading>",
    "satsInView": "<Number of satellites in view>",
    "satsUsed": "<Number of satellites in use>"
  }
}
```

get_imu_data

Request and clear the IMU measurement buffer, which contains either the most recent 1 second of IMU measurements or all of the measurements since this command was last called. If return_calibrated_data is explicitly set to false or not provided, this command will return the accumulated uncalibrated/raw measurements without resetting the calibrated measurement buffer. If return_calibrated_data is set to true, it will retrieve and reset the calibrated measurement buffer without resetting the raw measurement buffer.

This command is available with authorization of the following access role(s):

- SUPER_USER

User Request Parameters

Name	Description	Type	Constraints
return_calibrated_da ta	**Optional** True will return calibrated accelerometer data, false will return raw accelerometer measurements.	boolean	[True, False]

Radar Response Parameters

Name	Description	Type	Constraints
imu_data		*imu_data*	
data_is_calibrated	True if the output data is calibrated with onboard coefficients. Should match the input calibration boolean.	boolean	[True, False]
accel_cal_coefficien ts_exist	True if calibration coefficients exist for this unit's internal accelerometer.	boolean	[True, False]

imu_data

Name	Description	Type	Constraints
total	Total number of IMU measurements in this response.	integer	'uint16_t' range
imu_measurement_arra y	Array of IMU measurements.	*imu_mea*[]	Maximum 200 elements

imu_measurement

Name	Description	Type	Constraints
systime_ns	System time in nanoseconds that the measurement was recorded.	integer	'uint64_t' range
accel_x_mg	Accelerometer X-axis measurement in milli-gs.	number	'float' range
accel_y_mg	Accelerometer Y-axis measurement in milli-gs.	number	'float' range
accel_z_mg	Accelerometer Z-axis measurement in milli-gs.	number	'float' range
gyro_x_dps	Gyroscope x-axis measurement in degrees per second.	number	'float' range
gyro_y_dps	Gyroscope y-axis measurement in degrees per second.	number	'float' range
gyro_z_dps	Gyroscope z-axis measurement in degrees per second.	number	'float' range

Example Request

RadarIO Prompt Syntax

```
get_imu_data return_calibrated_data False
```

RadarIO Script Syntax

```
client.get_imu_data(return_calibrated_data=False)
```

JSON Syntax (as written to control port)

```
{
  "method": "get_imu_data",
  "id": 19,
  "params": {"return_calibrated_data": false}
}
```

Example Response

```
{
  "id": 19,
  "result": {
    "imu_data": "<imu_data>",
    "data_is_calibrated": "<True if the output data is calibrated with onboard␣
↪coefficients. Should match the input calibration boolean. >",
    "accel_cal_coefficients_exist": "<True if calibration coeffiecients exist␣
↪for this unit's internal accelerometer. >"
  }
}
```

3.8 IBIT Commands

Commands for initiating built in tests

- *get_ibit_dram_perf_results*
- *ibit_antenna_diode_check*
- *ibit_antenna_loopback_check*
- *ibit_dram_perf*
- *ibit_dram_test*
- *ibit_fsck*
- *ibit_loopback*
- *ibit_ramhog*
- *ibit_unleash_watchdog*

get_ibit_dram_perf_results

Read back results of *ibit_dram_perf* command

This command is available with authorization of the following access role(s):

- SUPER_USER

Radar Response Parameters

Name	Description	Type	Constraints
ddr_apm_txt	Contents of *ibit_dram_perf* results file.	string	ASCII characters

Example Request

RadarIO Prompt Syntax

```
get_ibit_dram_perf_results
```

RadarIO Script Syntax

```
client.get_ibit_dram_perf_results()
```

JSON Syntax (as written to control port)

```
{"method": "get_ibit_dram_perf_results", "id": 48}
```

Example Response

```
{
  "id": 48,
  "result": {"ddr_apm_txt": "<Contents of 'ibit_dram_perf' results file.>"}
}
```

ibit_antenna_diode_check

Execute an ibit to run the antenna diode check. The system must be in the service state (see *enter_service* command) to ibit_antenna_diode_check.

This command is available with authorization of the following access role(s):

- SUPER_USER

Radar Response Parameters

Name	Description	Type	Constraints
faults	17 columns of rows of drivers	integer[17][8	$0 \leq x \leq 4$

Example Request

RadarIO Prompt Syntax

```
ibit_antenna_diode_check
```

RadarIO Script Syntax

```
client.ibit_antenna_diode_check()
```

JSON Syntax (as written to control port)

```
{"method": "ibit_antenna_diode_check", "id": 51}
```

Example Response

```
{"id": 51, "result": {"faults": "<17 columns of rows of drivers>"}}
```

ibit_antenna_loopback_check

The loopback test checks each data line (0-3) for each row (0-7) ensuring the output of the row, when looped back is correct. This ensures that the dataline trace through the entire row is not broken, is propagating correctly through the ASICS, and is not stuck high or low. It also provides some test coverage for the BIT line connectivity between the BIT MUXES back to the MAPCOM/SoC. FAULT_NONE = 0, STUCK_HIGH = 1, STUCK_LOW = 2

This command is available with authorization of the following access role(s):

- SUPER_USER

Radar Response Parameters

Name	Description	Type	Constraints
faults	8 cols of rows	integer[8][4]	$0 \leq x \leq 4$

Example Request

RadarIO Prompt Syntax

```
ibit_antenna_loopback_check
```

RadarIO Script Syntax

```
client.ibit_antenna_loopback_check()
```

JSON Syntax (as written to control port)

```
{"method": "ibit_antenna_loopback_check", "id": 52}
```

Example Response

```
{"id": 52, "result": {"faults": "<8 cols of rows>"}}
```

ibit_dram_perf

Execute an ibit command to test DRAM throughput

This command is available with authorization of the following access role(s):

- SUPER_USER

User Request Parameters

Name	Description	Type	Constraints
readings	Number of readings to perform	integer	$1 \leq x \leq 1000$
cpu_core	**Optional** Which CPU core to use	integer	$0 \leq x \leq 3$
inject	**Optional** Inject number	boolean	[True, False]
priority	**Optional** Process priority	integer	$0 \leq x \leq 99$

Example Request

RadarIO Prompt Syntax

```
ibit_dram_perf readings 10 cpu_core 1 inject True priority 2
```

RadarIO Script Syntax

```
client.ibit_dram_perf(readings=10, cpu_core=1, inject=True, priority=2)
```

JSON Syntax (as written to control port)

```
{
  "method": "ibit_dram_perf",
  "id": 47,
  "params": {"readings": 10, "cpu_core": 1, "inject": true, "priority": 2}
}
```

Example Response

```
{"id": 47, "result": {}}
```

ibit_dram_test

Execute test on SOM PL/PS DRAM

This command is available with authorization of the following access role(s):

- SUPER_USER

User Request Parameters

Name	Description	Type	Constraints
subsystem	SOM subsystem selection	enum	• SOM_PL • SOM_PS

Example Request

RadarIO Prompt Syntax

```
ibit_dram_test subsystem SOM_PL
```

RadarIO Script Syntax

```
client.ibit_dram_test(subsystem=SOM_PL)
```

JSON Syntax (as written to control port)

```
{"method": "ibit_dram_test", "id": 54, "params": {"subsystem": "SOM_PL"}}
```

Example Response

```
{"id": 54, "result": {}}
```

ibit_fsck

Get fsck results from last boot (always runs and repairs)

This command is available with authorization of the following access role(s):

- SUPER_USER

Radar Response Parameters

Name	Description	Type	Constraints
output	The output after reboot	string[]	ASCII characters Maximum 32 elements

Example Request

RadarIO Prompt Syntax

```
ibit_fsck
```

RadarIO Script Syntax

```
client.ibit_fsck()
```

JSON Syntax (as written to control port)

```
{"method": "ibit_fsck", "id": 49}
```

Example Response

```
{"id": 49, "result": {"output": "<The output after reboot>"}}
```

ibit_loopback

Set loopback point

This command is available with authorization of the following access role(s):

- SUPER_USER

User Request Parameters

Name	Description	Type	Constraints
point	**Optional** Loopback point	enum	• NONE • RF • DAC

Radar Response Parameters

Name	Description	Type	Constraints
point	Loopback point	enum	• NONE • RF • DAC

Example Request

RadarIO Prompt Syntax

```
ibit_loopback point DAC
```

RadarIO Script Syntax

```
client.ibit_loopback(point=DAC)
```

JSON Syntax (as written to control port)

```
{"method": "ibit_loopback", "id": 50, "params": {"point": "DAC"}}
```

Example Response

```
{"id": 50, "result": {"point": "<Loopback point>"}}
```

ibit_ramhog

Execute RAM saturation test using specified CPU core

This command is available with authorization of the following access role(s):

- SUPER_USER

User Request Parameters

Name	Description	Type	Constraints
cpu_core	**Optional** Which CPU core to use	integer	$0 \leq x \leq 3$
MB	**Optional** Number of megabytes to hog	integer	$1 \leq x \leq 1024$
priority	**Optional** Process priority	integer	$0 \leq x \leq 99$
type	**Optional** Ramhog data access type	enum	• READ_ONLY • WRITE_ONLY • READ_WRITE
stop	**Optional** Use to stop the test	boolean	[True, False]

Example Request

RadarIO Prompt Syntax

```
ibit_ramhog cpu_core 1 MB 195 priority 59 type READ_WRITE stop True
```

RadarIO Script Syntax

```
client.ibit_ramhog(cpu_core=1, MB=195, priority=59, type=READ_WRITE, stop=True)
```

JSON Syntax (as written to control port)

```
{
  "method": "ibit_ramhog",
  "id": 46,
  "params": {
    "cpu_core": 1, "MB": 195, "priority": 59, "type": "READ_WRITE", "stop": true
```

(continues on next page)

(continued from previous page)

```
  }
}
```

Example Response

```
{"id": 46, "result": {}}
```

ibit_unleash_watchdog

Stop servicing the watchdog timer to trigger a watchdog system reset

This command is available with authorization of the following access role(s):

- SUPER_USER

Example Request

RadarIO Prompt Syntax

```
ibit_unleash_watchdog
```

RadarIO Script Syntax

```
client.ibit_unleash_watchdog()
```

JSON Syntax (as written to control port)

```
{"method": "ibit_unleash_watchdog", "id": 53}
```

Example Response

```
{"id": 53, "result": {}}
```

3.9 Glossary of Enumerations

A handful of command parameters expect one of several defined values. This section lists each of these enumerated types, with definitions for each value.

ROLE_LEVELS

Table 84: ROLE_LEVELS Definitions

Value	Description
USER	Authorized by default, covers all API necessary to set up and run the radar.
SUPER_USER	Covers configuration commands/parameters for fine-tuning of radar operation.
FIELD_UPGRADE_USER	Covers all commands related to field-upgrades of the radar firmware.
ECHODYNE_ONLY	Provides access to all API.

LOG_CATEGORIES

Table 85: LOG_CATEGORIES Definitions

Value	Description
GENERIC	Logs that don't relate to a particular worker state machine.
SYS_MGR	Logs relating to the 'System Manager' state machine.
CONFIG_MGR	Logs relating to the 'Configuration Manager' state machine.
FILE_MGR	Logs relating to the 'File Manager' state machine.
CMD_MGR	Logs relating to the 'Command Manager' state machine.
OUTPUT_MGR	Logs relating to the 'Output Manager' state machine.
DATA_CONN	Logs relating to data 'Data Conn' state machine.
HW_MGR	Logs relating to the 'Hardware Manager' state machine.
MASTER_CTRL	Logs relating to the 'Master Control' state machine.
EXCITER_CTRL	Logs relating to the 'Exciter Control' state machine.
TX0_CTRL	Logs relating to the 'Transmit Control 0' state machine.
TX1_CTRL	Logs relating to the 'Transmit Control 1' state machine.
FPGA_CTRL	Logs relating to the 'FPGA Control' state machine.
ZYNQ_FPGA	Logs relating to the 'Zynq FPGA' Accessor.
RADAR_MGR	Logs relating to the 'Radar Manager' state machine.
TRACKER	Logs relating to the 'Tracker' state machine.
DETECTOR	Logs relating to the 'Detector' state machine.
BEAM_SCHED	Logs relating to the 'Beam Scheduler' state machine.
CLASSIFIER	Logs relating to the 'Classifier' state machine.
CALIBRATOR	Logs relating to the 'Calibrator' state machine.
STATE_MACH_TEST	Logs relating to the 'State Machine Test' state machine.
MEMPOOL	Logs relating to process memory pools.
DRIVER	Logs relating to device drivers.
I2C	Logs relating to internal I2C transfers.
SPI	Logs relating to internal SPI transfers.
UNIT_TEST	Logs relating to unit testing.
PERF_MEAS	Logs relating to performance measurements
ADAPTER	Logs relating to system interrupts and forks/subprocesses.
CMD_PARSE	Logs relating to parsing of incoming commands on the command port.
BSP	Logs relating to the Board Support Package interface.
STAGE_FILE	Logs relating to field-upgrade file uploading/staging.
AD9081	Logs relating to the AD9081 driver.
FPGA_DMA	Logs relating to FPGA DMA buffer access.
RESET_BTN	Logs relating to reset button presses.
SYS_CFG	Logs relating to 'System Configuration' test-maps.
SYS_BKGD	Logs relating to the 'System Background' worker.
FPGA_BEAM_STATS	Logs providing beam statistics.
VDOM_CTRL	Logs relating to the 'VDOM Control' state machine.
BIT	Logs relating to built-in tests.

continues on next page

Table 85 – continued from previous page

Value	Description
FRAMEWORK	Logs relating to software framework.
ALWAYS	Generic logs that always pass log filtering.
RESERVED	N/A

LOG_LEVELS

Table 86: LOG_LEVELS Definitions

Value	Description
debug	Highly verbose debugging information / diagnostics helpful for developers.
info	General information providing insight into the state of the system.
notice	Logs regarding normal, but significant system conditions.
warning	Indications of off-nominal, but recoverable system conditions.
error	Indications of conditions that result in failure of some operation.
critical	Indications of serious issues that may lead to a system shutdown.

RADAR_MODES

Table 87: RADAR_MODES Definitions

Value	Description
Resetting	System is preparing to reset.
Reset	System has reset.
Initializing	System is initializing.
Operation	System has finished initializing and is now operable.
Operation:IDLE	System is idle
Operation:TX_ACTIVE	Antenna is energized and sensors are active, output data is being generated.
Calibration:TX_ACTIVE	Antenna is energized in calibration mode.
Service	All configuration parameters can be modified, system is non-operable.
Fault	An error has occurred, system is non-operable
Standby	System is in transition from . . . :TX_ACTIVE to Operation:IDLE

PLATFORMS

Table 88: PLATFORMS Definitions

Value	Description
iWave	iWave test platform
MAPCOM2	EchoShield hardware
x86	PC test platform

4 Parameters

This section describes the set of Radar Parameters used by the API to configure the Radar.

Each set of Parameters is linked to a Group which exposes a single command to modify the Parameters belonging to that Group. There is also a single command per Group used to query the current value of the Parameters in that Group.

- System Parameters can be set using the *set_sys* command and queried using the *get_sys* command.
- Network Parameters can be set using the *set_net* command and queried using the *get_net* command.
- RadarApp Parameters can be set using the *set* command and queried using the *get* command.

4.1 Parameter Attributes

Each parameter in this section is described by a set of attributes, detailed below:

Default

Factory default value of the parameter.

Type

JSON type of the parameter.

Constraints

Constraints that define the range of acceptable values for the parameter. Constraints of *None* indicates that the parameter will accept any value appropriate for its type.

Nullable

Whether the parameter can be set to *null* or *None*.

Modifiable Mode

Indicates which mode the radar must be in to set the parameter value. The two applicable modes are:

- **Operation** - Default mode
- **Service** - Mode for adjusting radar configuration. Can be entered and exited with the commands *enter_service* and *exit_service*.

Some modifiable parameters can only be modified while the radar is in Service mode. The rest can be modified either in Operating or Service states.

Some parameters are not modifiable, these will display a Modifiable Mode of "Read Only".

The radar will respond to 'set' commands with a STATE error if attempting to set a parameter that cannot be set in the current mode.

Persistence

Describes what operations will reset the parameter to its factory default value.

- **Persistent** Parameters can only be reset using the Factory reset process.
- **Sticky** Parameters can be reset by parameter group to their default values using the associated reset command, see available reset commands *below*.
- **Volatile** Parameters will be reset to their default value on the next system boot.

	Parameter Type		
Reset Operation	**Persistent**	**Sticky**	**Volatile**
Power Cycle	No	No	Yes
Software Reset	No	Yes	Yes
Factory Reset	Yes	Yes	Yes

Parameter Group	Reset Command
RadarApp	*reset_radar_params*

User Access

Lists user access roles that must be authorized with the *authorize_role* command in order to modify the parameter. See the *Role Authentication* section.

4.2 System Parameters

led_indicators_enabled

Boolean representing whether the LED indicators are enabled.

Attribute	Value
Default	True
Type	boolean
Constraints	[True, False]
Nullable	No
Modifiable Mode	Service
Persistence	Persistent
User Access	USER (default)

logcat_list

A string containing a list of space or comma separated log categories to enable

Attribute	Value
Default	“”
Type	string
Constraints	ASCII characters
Nullable	No
Modifiable Mode	Service
Persistence	Persistent
User Access	SUPER_USER

loglevel

Settable logging levels

Attribute	Value
Default	"warning"
Type	enum
Constraints	• debug • info • notice • warning • error • critical
Nullable	No
Modifiable Mode	Service
Persistence	Persistent
User Access	SUPER_USER

persist_logs

If True, system logs are persisted on disk. If False, system logs are not persisted on disk. Syslog streaming remains active regardless of this parameter's value. Existing persisted logs remain when this parameter's value is changed, see the *delete_existing_logs* command to remove all existing persisted logs.

Attribute	Value
Default	True
Type	boolean
Constraints	[True, False]
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Persistent
User Access	SUPER_USER

pps_source

Pulse Per Second (PPS) Source

Attribute	Value
Default	"PPS_SOURCE_GPS"
Type	enum
Constraints	• PPS_SOURCE_GPS • PPS_SOURCE_USER • PPS_SOURCE_EMULATED • PPS_SOURCE_TEST
Nullable	No
Modifiable Mode	Service
Persistence	Persistent
User Access	USER (default)

status_packet_period

Seconds between status update packets.

Attribute	Value
Default	1
Type	integer
Constraints	$1 \leq x \leq 60$
Nullable	No
Modifiable Mode	Service
Persistence	Persistent
User Access	USER (default)

4.3 Network Parameters

address_10G

10Gb/s ethernet interface IPv4 address

Attribute	Value
Default	"172.16.10.20"

continues on next page

Table 95 – continued from previous page

Attribute	Value
Type	string
Constraints	ASCII characters
Nullable	Yes
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

address_1G

1Gb/s ethernet interface IPv4 address

Attribute	Value
Default	“192.168.1.150”
Type	string
Constraints	ASCII characters
Nullable	No
Modifiable Mode	Service
Persistence	Persistent
User Access	USER (default)

beams_port

1Gb/s ethernet interface beams data port

Attribute	Value
Default	29986
Type	integer
Constraints	$1024 \leq x \leq 49151$
Nullable	No
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

beams_schema_generation

Update the schema generation ID for beams packets

Attribute	Value
Default	"gen3"
Type	enum
Constraints	• gen3
Nullable	No
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

command_port

1Gb/s ethernet interface IPv4 command port

Attribute	Value
Default	29978
Type	integer
Constraints	$1024 \leq x \leq 49151$
Nullable	No
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

cookies_port

1Gb/s ethernet interface cookie data port

Attribute	Value
Default	29985
Type	integer
Constraints	$1024 \leq x \leq 49151$
Nullable	No
Modifiable Mode	Service
Persistence	Sticky

continues on next page

Table 100 – continued from previous page

Attribute	Value
User Access	USER (default)

cookies_schema_generation

Update the schema generation ID for cookies packets

Attribute	Value
Default	“gen3”
Type	enum
Constraints	• gen3
Nullable	No
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

detections_port

1Gb/s ethernet interface detection data port

Attribute	Value
Default	29981
Type	integer
Constraints	$1024 \leq x \leq 49151$
Nullable	No
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

detections_schema_generation

Update the schema generation ID for detections packets

Attribute	Value
Default	“gen3”
Type	enum
Constraints	• gen3
Nullable	No
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

dhcp_1G

Allows DHCP to be disabled, enabled, or depend on a HW select

Attribute	Value
Default	“HW_SELECT”
Type	enum
Constraints	• DISABLE • ENABLE • HW_SELECT
Nullable	No
Modifiable Mode	Service
Persistence	Persistent
User Access	USER (default)

dns_primary_1G

1Gb/s ethernet interface IPv4 primary DNS server

Attribute	Value
Default	None
Type	string

continues on next page

Table 105 – continued from previous page

Attribute	Value
Constraints	ASCII characters
Nullable	Yes
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

dns_secondary_1G

1Gb/s ethernet interface IPv4 secondary DNS server

Attribute	Value
Default	None
Type	string
Constraints	ASCII characters
Nullable	Yes
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

gateway_10G

10Gb/s ethernet interface IPv4 default gateway

Attribute	Value
Default	“172.16.10.1”
Type	string
Constraints	ASCII characters
Nullable	Yes
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

gateway_1G

1Gb/s ethernet interface IPv4 default gateway

Attribute	Value
Default	“192.168.1.1”
Type	string
Constraints	ASCII characters
Nullable	No
Modifiable Mode	Service
Persistence	Persistent
User Access	USER (default)

gps_data_rate

GPS data polling rate

Attribute	Value
Default	“GPS_RATE_1HZ”
Type	enum
Constraints	• GPS_RATE_1HZ • GPS_RATE_10HZ
Nullable	No
Modifiable Mode	Service
Persistence	Persistent
User Access	USER (default)

gps_time_disable

If true, disable gps as a time source even if it is available.

Attribute	Value
Default	False
Type	boolean
Constraints	[True, False]
Nullable	No
Modifiable Mode	Service

continues on next page

Table 110 – continued from previous page

Attribute	Value
Persistence	Persistent
User Access	USER (default)

hostname_1G

1Gb/s ethernet interface IPv4 host name

Attribute	Value
Default	“echoshield-01”
Type	string
Constraints	ASCII characters
Nullable	No
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

imu_data_rate

IMU data polling rate

Attribute	Value
Default	“IMU_LOW_RATE”
Type	enum
Constraints	• IMU_LOW_RATE • IMU_HIGH_RATE
Nullable	No
Modifiable Mode	Service
Persistence	Persistent
User Access	USER (default)

keep_alive_count

Number of unacknowledged keep-alive probes before determining a connection on a socket is dead. Probes are sent once per second. NOTE: The new keep_alive_count value will only apply to new socket connections.

Attribute	Value
Default	3
Type	integer
Constraints	$1 \leq x \leq 100$
Nullable	No
Modifiable Mode	Service
Persistence	Sticky
User Access	SUPER_USER

kinematics_input_port

1Gb/s ethernet interface IPv4 UDP kinematics input stream port

Attribute	Value
Default	29989
Type	integer
Constraints	$1024 \leq x \leq 49151$
Nullable	No
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

kinematics_port

1Gb/s ethernet interface kinematics data packet output port

Attribute	Value
Default	29988
Type	integer
Constraints	$1024 \leq x \leq 49151$
Nullable	No
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

kinematics_schema_generation

Update the schema generation ID for kinematics packets

Attribute	Value
Default	"gen3"
Type	enum
Constraints	• gen3
Nullable	No
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

link_speed_override

Override the low speed behavior on the 1GbE interface, which disallows connection to certain data ports when the negotiated link speed is lower than 1Gbps.

Attribute	Value
Default	False
Type	boolean
Constraints	[True, False]
Nullable	No
Modifiable Mode	Service
Persistence	Volatile
User Access	SUPER_USER

locator_port

1Gb/s ethernet interface IPv4 locator port

Attribute	Value
Default	29977
Type	integer
Constraints	$1024 \leq x \leq 49151$
Nullable	No
Modifiable Mode	Service

continues on next page

Table 118 – continued from previous page

Attribute	Value
Persistence	Sticky
User Access	USER (default)

measurements_port

1Gb/s ethernet interface measurement data port

Attribute	Value
Default	29984
Type	integer
Constraints	$1024 \leq x \leq 49151$
Nullable	No
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

measurements_schema_generation

Update the schema generation ID for measurements packets

Attribute	Value
Default	“gen3”
Type	enum
Constraints	• gen3
Nullable	No
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

netmask_10G

10Gb/s ethernet interface IPv4 subnet mask

Attribute	Value
Default	“255.255.255.0”
Type	string
Constraints	ASCII characters
Nullable	Yes
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

netmask_1G

1Gb/s ethernet interface IPv4 subnet mask

Attribute	Value
Default	“255.255.255.0”
Type	string
Constraints	ASCII characters
Nullable	No
Modifiable Mode	Service
Persistence	Persistent
User Access	USER (default)

ntp_primary_1G

1Gb/s ethernet interface IPv4 NTP primary server

Attribute	Value
Default	None
Type	string
Constraints	ASCII characters
Nullable	Yes
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

ntp_secondary_1G

1Gb/s ethernet interface IPv4 NTP secondary server

Attribute	Value
Default	None
Type	string
Constraints	ASCII characters
Nullable	Yes
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

ntp_tertiary_1G

1Gb/s ethernet interface IPv4 NTP tertiary server

Attribute	Value
Default	None
Type	string
Constraints	ASCII characters
Nullable	Yes
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

rvmap_address_10G

10Gb/s ethernet interface IPv4 RVmap UDP destination address

Attribute	Value
Default	“172.16.10.10”
Type	string
Constraints	ASCII characters
Nullable	Yes
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

rvmap_port_10G

10Gb/s ethernet interface IPv4 Rvmap UDP destination port

Attribute	Value
Default	30980
Type	integer
Constraints	$1024 \leq x \leq 49151$
Nullable	No
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

search_domain_1G

1Gb/s ethernet interface IPv4 DNS search domain name

Attribute	Value
Default	“”
Type	string
Constraints	ASCII characters
Nullable	No
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

status_port

1Gb/s ethernet interface status data port

Attribute	Value
Default	29979
Type	integer
Constraints	$1024 \leq x \leq 49151$
Nullable	No
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

status_schema_generation

Update the schema generation ID for status packets

Attribute	Value
Default	“gen3”
Type	enum
Constraints	• gen3
Nullable	No
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

syslog_host_1G

IPv4 of machine configured to receive syslog messages

Attribute	Value
Default	“192.168.1.1”
Type	string
Constraints	ASCII characters
Nullable	Yes
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

syslog_port_1G

Port configured to receive syslog messages

Attribute	Value
Default	5514
Type	integer
Constraints	$10 \leq x \leq 49151$
Nullable	No
Modifiable Mode	Service
Persistence	Sticky

continues on next page

Table 132 – continued from previous page

Attribute	Value
User Access	USER (default)

track_handoff_input_port

1Gb/s ethernet interface IPv4 UDP track handoff input stream port

Attribute	Value
Default	29990
Type	integer
Constraints	$1024 \leq x \leq 49151$
Nullable	No
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

tracks_extended

Set tracks output packet to extended mode

Attribute	Value
Default	False
Type	boolean
Constraints	[True, False]
Nullable	No
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

tracks_output_rate_hz

Sets the maximum rate a given track is output to the user. If tracks_output_rate_hz >= track_update_rate_hz, tracks will be reported to the user at track_update_rate_hz. When tracks_output_rate_hz < track_update_rate_hz, tracks will be reported to the user when more than 1/tracks_output_rate_hz seconds have ellapsed since the last track data was reported to the user and a track update has occured. This means the true track output rate can be slightly less than the configured track output rate. NOTE: When tracks_output_rate_hz < track_update_rate_hz, it is no longer possible to find all measurements that have been associated to a track which can impact certain data analysis activities. NOTE: This setting is super-

seded by tracks_output_report_all_tracks_enable. If tracks_output_report_all_tracks_enable = true, tracks will be output to the user at track_update_rate_hz.

Attribute	Value
Default	10.0
Type	number
Constraints	$0.1 \leq x \leq 10.0$
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Volatile
User Access	USER (default)

tracks_output_report_all_tracks_enable

When enabled, this overrides all track output filters and output reduction schemes causing all track updates to be reported to the user. Setting this to true allows Echodyne engineers to collect all data needed to perform analysis on collected data.

Attribute	Value
Default	False
Type	boolean
Constraints	[True, False]
Nullable	No
Modifiable Mode	Service
Persistence	Volatile
User Access	USER (default)

tracks_port

1Gb/s ethernet interface track data port

Attribute	Value
Default	29982
Type	integer
Constraints	$1024 \leq x \leq 49151$
Nullable	No
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

tracks_schema_generation

Update the schema generation ID for track packets

Attribute	Value
Default	"gen3"
Type	enum
Constraints	• gen3
Nullable	No
Modifiable Mode	Service
Persistence	Sticky
User Access	USER (default)

4.4 RadarApp Parameters

Some RadarApp Parameter defaults are based on the active mission. See the *configure_mission* and *list_missions* API commands.

All parameters that change their default value based on the active mission are listed in the matrix below:

Parameter \ Active Mission	Default	C-UAS_1	C-UAS_2	COASTAL_CUAS	DISMOUNT
detection_clutter_max_ang…	6	—	—	—	90.0
detection_clutter_vr_spre…	0.5	—	—	—	0.25
detection_dynamic_clutter…	False	—	—	True	—
dsp_cfar_beam_pfa	0.01	—	—	—	0.1
fov_seg_max_target_roi_m	8000	—	15000.0	15000.0	15000.0
fov_seg_max_target_up_of_…	1000	—	2000.0	2000.0	50.0
fov_seg_min_target_roi_m	150	—	300.0	300.0	500.0
target_max_agl_m	4000	—	5000.0	5000.0	10.0
target_max_lateral_accele…	5	—	10.0	10.0	2.0
target_max_length_m	1.0	—	3.0	3.0	2.0
target_max_velocity_m_s	35	—	70.0	70.0	30.0
target_max_vertical_accel…	2	—	3.5	3.5	0.5
target_min_agl_m	20	—	40.0	40.0	-10.0
track_confirmations	1	—	—	—	3
track_max_coast_time_s	5	—	—	—	10.0
track_max_confidence_time…	1	—	—	—	2.0

detection_clutter_max_angle_above_ground_deg

The maximum elevation angle of a detection in degrees, relative to ground elevation angle, at which the stationary clutter mask will be applied. Detections above this elevation angle and above detection_clutter_max_height_above_ground_m will not be masked even if they have a radial velocity that falls within the specified stationary clutter region. If there are many undesired tracks that correspond to stationary objects with a reported elevation angle above ground greater than detection_clutter_max_angle_above_ground_deg, this parameter can be increased to help remove those tracks.

Active Mission	Default	C-UAS_1	C-UAS_2	COASTAL_C-UAS	DISMOUNT
Default Value	6	—	—	—	90.0

Attribute	Value
Type	number
Constraints	$0 \leq x \leq 90$
Nullable	No
Modifiable Mode	Service and Operation
Persistence	Volatile
User Access	SUPER_USER

detection_clutter_max_height_above_ground_m

The maximum height of a detection in meters, relative to the ground, at which the stationary clutter mask will be applied. Detections above this height and above detection_clutter_max_angle_above_ground_deg will not be masked even if they have a radial velocity that falls within the specified stationary clutter region. If there are many undesired tracks that correspond to stationary objects with a reported height above ground greater than detection_clutter_max_height_above_ground_m, this parameter can be increased to help remove those tracks.

Attribute	Value
Default	100
Type	number
Constraints	$-1000000 \leq x \leq 1000000$
Nullable	No
Modifiable Mode	Service and Operation
Persistence	Volatile
User Access	SUPER_USER

detection_clutter_vr_spread_mps

The real radial velocity spread of stationary clutter, in units of meters per second. Increasing this value will decrease the likelihood things such as trees swaying in the wind will generate tracks, but will also decrease the radar's ability to track slow moving targets in the region defined by detection_clutter_max_height_above_ground_m and detection_clutter_max_angle_above_ground_deg.

Active Mission	Default	C-UAS_1	C-UAS_2	COASTAL_C-UAS	DISMOUNT
Default Value	0.5	—	—	—	0.25

Attribute	Value
Type	number
Constraints	$0 \leq x \leq 1000000$
Nullable	No
Modifiable Mode	Service and Operation
Persistence	Volatile
User Access	SUPER_USER

detection_dynamic_clutter_hough_threshold

The minimum clutter score required for any detections to be masked by the dynamic clutter mask. Increasing this value will reduce the amount of clutter masked.

Attribute	Value
Default	35.0
Type	number
Constraints	$1.0 \leq x \leq 10000.0$
Nullable	No
Modifiable Mode	Service and Operation
Persistence	Volatile
User Access	SUPER_USER

detection_dynamic_clutter_mask_enable

Flag representing whether to enable/disable the dynamic clutter mask. This mask can be used to mask ground clutter with a non zero mean velocity such as ocean waves.

Active Mission	Default	C-UAS_1	C-UAS_2	COASTAL_C-UAS	DISMOUNT
Default Value	False	—	—	True	—

Attribute	Value
Type	boolean
Constraints	[True, False]
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Volatile
User Access	SUPER_USER

detection_dynamic_clutter_max_angle_above_ground_deg

The maximum elevation angle of a detection, relative to ground elevation angle, at which the dynamic ground clutter mask will be applied. Detections above this elevation angle and above detection_dynamic_clutter_max_height_above_ground_m will not be masked even if they have a radial velocity that matches the indentified dynamic ground clutter.

Attribute	Value
Default	4
Type	number
Constraints	$0 \leq x \leq 90$
Nullable	No
Modifiable Mode	Service and Operation
Persistence	Volatile
User Access	SUPER_USER

detection_dynamic_clutter_max_height_above_ground_m

The maximum height of a detection, relative to the ground, at which the dynamic ground clutter mask will be applied. Detections above this height and above detection_dynamic_clutter_max_angle_above_ground_deg will not be masked even if they have a radial velocity that matches the indentified dynamic ground clutter.

Attribute	Value
Default	60
Type	number
Constraints	$-1000000 \leq x \leq 1000000$
Nullable	No
Modifiable Mode	Service and Operation
Persistence	Volatile
User Access	SUPER_USER

dsp_cfar_beam_pfa

Probability a given beam will produce a false alarm detection. Increasing the PFA value will increase the likelihood that a real object will be detected, but also increases the likelihood a detection is from noise or clutter. The PFA should be calibrated depending on user's environment and needs. A low PFA is recommended for use cases where the user wants to avoid false detections and tracks at all costs while a slightly higher PFA is recommended where clutter is limited and detecting objects at maximum range is critical.

Active Mission	Default	C-UAS_1	C-UAS_2	COASTAL_C-UAS	DISMOUNT
Default Value	0.01	—	—	—	0.1

Attribute	Value
Type	number
Constraints	$1e-08 \leq x \leq 1$
Nullable	No
Modifiable Mode	Service and Operation
Persistence	Volatile
User Access	SUPER_USER

fov_seg_max_target_roi_m

The maximum range at which the system is capable of detecting and tracking a target. This parameter is also used to set the maximum RDmap subsection range. As fov_seg_max_target_roi_m is increased, the system will have less time to observe targets at closer ranges.

Active Mission	Default	C-UAS_1	C-UAS_2	COASTAL_C-UAS	DISMOUNT
Default Value	8000	—	15000.0	15000.0	15000.0

Attribute	Value
Type	number
Constraints	$25 \leq x \leq 25000$
Nullable	No
Modifiable Mode	Service
Persistence	Volatile
User Access	SUPER_USER

fov_seg_max_target_up_of_interest_m

The maximum height in meters above the radar where a user wishes to detect a target to max range. This parameter is used for waveform field-of-view segmentation. As fov_seg_max_target_up_of_interest_m is increased, field-of-view scan times will also increase.

Active Mission	Default	C-UAS_1	C-UAS_2	COASTAL_C-UAS	DISMOUNT
Default Value	1000	—	2000.0	2000.0	50.0

Attribute	Value
Type	number
Constraints	$0 \leq x \leq 20000$
Nullable	No
Modifiable Mode	Service
Persistence	Volatile
User Access	SUPER_USER

fov_seg_min_target_roi_m

This parameter is used to help automatically constrain the system's search field of view. If a given beam direction fully intersects the ground or the ceiling as defined by the local terrain and fov_seg_max_target_up_of_interest_m respectively, at a range less than fov_seg_min_target_roi_m, the system will not execute search beams in that direction. fov_seg_min_target_roi_m should be set to the minimum range at which the system is expected to acquire track on new targets.

Active Mission	Default	C-UAS_1	C-UAS_2	COASTAL_C-UAS	DISMOUNT
Default Value	150	—	300.0	300.0	500.0

Attribute	Value
Type	number
Constraints	$100 \leq x \leq 25000$
Nullable	No
Modifiable Mode	Service
Persistence	Volatile
User Access	SUPER_USER

meas_clutter_vr_spread_mps

The real radial velocity spread of stationary clutter, in units of meters per second. Measurements with a radial velocity less than meas_clutter_vr_spread_mps from the expected radial velocity of a world stationary object can not form a track. This is similar to the detection stationary clutter mask, but applies to all measurements and only prevents a track from forming, not persisting. It is recommend to set this to <= 1.5 meters per second if the radar is producing a significant number of stationary tracks that are in the air and do not correspond to any real targets.

Attribute	Value
Default	0.0
Type	number
Constraints	$0 \leq x \leq 1000000$
Nullable	No
Modifiable Mode	Service and Operation
Persistence	Volatile
User Access	SUPER_USER

meas_max_rcs_dBsm

Maximum measurement RCS in dBsm. Measurements with a RCS above this threshold are masked and not used for target tracking.

Attribute	Value
Default	100
Type	number
Constraints	$-100 \leq x \leq 100$
Nullable	No
Modifiable Mode	Service and Operation
Persistence	Volatile
User Access	SUPER_USER

meas_min_rcs_dBsm

Minimum measurement RCS in dBsm. Measurements with a RCS below this threshold are masked and not used for target tracking.

Attribute	Value
Default	-100
Type	number
Constraints	$-100 \leq x \leq 100$
Nullable	No
Modifiable Mode	Service and Operation
Persistence	Volatile
User Access	SUPER_USER

mission

Configurable mission sets. NOTE: The mission set should only be updated through the configure_mission command.

Attribute	Value
Default	“C-UAS_1”
Type	enum

continues on next page

Table 161 – continued from previous page

Attribute	Value
Constraints	• C-UAS_1 • C-UAS_2 • DISMOUNT • COASTAL_C-UAS
Nullable	No
Modifiable Mode	Client Read Only (FW Modifiable)
Persistence	Volatile
User Access	USER (default)

search_azmax

Maximum azimuth search beam angle in degrees. Special constraint: search_azmin < search_azmax.

Attribute	Value
Default	65
Type	number
Constraints	$-65 \leq x \leq 65$
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Volatile
User Access	USER (default)

search_azmin

Minimum azimuth search beam angle in degrees. Special constraint: search_azmin < search_azmax.

Attribute	Value
Default	-65
Type	number
Constraints	$-65 \leq x \leq 65$
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Volatile
User Access	USER (default)

search_beams_enable

Whether or not to execute search beams

Attribute	Value
Default	True
Type	boolean
Constraints	[True, False]
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Volatile
User Access	SUPER_USER

search_elmax

Maximum elevation search beam angle in degrees. Special constraint: search_elmin < search_elmax.

Attribute	Value
Default	50
Type	number
Constraints	$-40 \leq x \leq 50$
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Volatile
User Access	USER (default)

search_elmin

Minimum elevation search beam angle in degrees. When search_fov_optimization_enable = true, the radar automatically prevents search beams from being executed in beam directions where the radar does not expect to be able to detect any objects, such as when looking straight into the ground. This parameter does not need to be updated to optimize radar performance and should be left at default unless the user needs to explicitly define the radar's min search elevation for a reason unrelated to tracking performance. Special constraint: search_elmin < search_elmax.

Attribute	Value
Default	-40
Type	number
Constraints	$-40 \leq x \leq 50$

continues on next page

Table 166 – continued from previous page

Attribute	Value
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Volatile
User Access	SUPER_USER

search_fov_optimization_enable

When enabled, the system will automatically skip search beams with a max line of sight less than or equal to the system's blind range. A user should only set this to false if they need the system to search over the entire defined search field-of-view as it will cause the system's overall search rate to decrease.

Attribute	Value
Default	True
Type	boolean
Constraints	[True, False]
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Volatile
User Access	SUPER_USER

target_max_agl_m

The maximum height above ground for a target of interest in meters. This is used to determine which tracks to allocate active tracking beam resources for. It is recommended to make sure target_min_agl_m < target_max_agl_m.

Active Mission	Default	C-UAS_1	C-UAS_2	COASTAL_C-UAS	DISMOUNT
Default Value	4000	—	5000.0	5000.0	10.0

Attribute	Value
Type	number
Constraints	$-100000 \leq x \leq 100000$
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Volatile

continues on next page

Table 169 – continued from previous page

Attribute	Value
User Access	SUPER_USER

target_max_lateral_acceleration_m_s2

The maximum lateral acceleration of a target of interest in meters per second squared. This is used to determine the process noise to add to tracks in the east & north direction.

Active Mission	Default	C-UAS_1	C-UAS_2	COASTAL_C-UAS	DISMOUNT
Default Value	5	—	10.0	10.0	2.0

Attribute	Value
Type	number
Constraints	$0 < x \leq 100$
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Volatile
User Access	SUPER_USER

target_max_length_m

The maximum physical extent of a target of interest in meters. This should be configured to the maximum dimension of any target a user wishes to track.

Active Mission	Default	C-UAS_1	C-UAS_2	COASTAL_C-UAS	DISMOUNT
Default Value	1.0	—	3.0	3.0	2.0

Attribute	Value
Type	number
Constraints	$0 < x \leq 100$
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Volatile

continues on next page

Table 173 – continued from previous page

Attribute	Value
User Access	SUPER_USER

target_max_velocity_m_s

The maximum velocity of a target of interest in meters per second. This should be configured to be approximately equal to the maximum speed of targets the user wishes to track. Note, a given mission set is not expected to support target velocities greater than approximately 3x the default target_max_velocity_m_s for that mission.

Active Mission	Default	C-UAS_1	C-UAS_2	COASTAL_C-UAS	DISMOUNT
Default Value	35	—	70.0	70.0	30.0

Attribute	Value
Type	number
Constraints	$0 < x \leq 1000$
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Volatile
User Access	SUPER_USER

target_max_vertical_acceleration_m_s2

The maximum vertical acceleration of a target of interest in meters per second squared. This is used to determine the process noise to add to tracks in the up direction.

Active Mission	Default	C-UAS_1	C-UAS_2	COASTAL_C-UAS	DISMOUNT
Default Value	2	—	3.5	3.5	0.5

Attribute	Value
Type	number
Constraints	$0 < x \leq 100$
Nullable	No

continues on next page

Table 177 – continued from previous page

Attribute	Value
Modifiable Mode	Operation and Service
Persistence	Volatile
User Access	SUPER_USER

target_min_agl_m

The minimum height above ground for a target of interest in meters. This is used to determine which tracks to allocate active tracking beam resources for. It is recommended to make sure target_min_agl_m < target_max_agl_m.

Active Mission	Default	C-UAS_1	C-UAS_2	COASTAL_C-UAS	DISMOUNT
Default Value	20	—	40.0	40.0	-10.0

Attribute	Value
Type	number
Constraints	$-100000 \leq x \leq 100000$
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Volatile
User Access	SUPER_USER

track_azmax

Tracking field-of-view maximum azimuth boundary in degrees. Tracks outside this range will not be updated by the radar and will coast for less than or equal to the Max Out of Bounds Coast Time (track_max_oob_coast_time_s) before being deleted by the tracker. Special constraint: track_azmin < track_azmax.

Attribute	Value
Default	65
Type	number
Constraints	$-65 \leq x \leq 65$
Nullable	No
Modifiable Mode	Operation and Service

continues on next page

Table 180 – continued from previous page

Attribute	Value
Persistence	Volatile
User Access	USER (default)

track_azmin

Tracking field-of-view minimum azimuth boundary in degrees. Tracks outside this range will not be updated by the radar and will coast for less than or equal to the Max Out of Bounds Coast Time (track_max_oob_coast_time_s) before being deleted by the tracker. Special constraint: track_azmin < track_azmax.

Attribute	Value
Default	-65
Type	number
Constraints	$-65 \leq x \leq 65$
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Volatile
User Access	USER (default)

track_confirmations

This parameter is responsible for determining how many times the radar needs to repeatedly detect an object in the same region before higher precision/cost sequential lobing beams are used to look back at the target and the track is considered confirmed. Keep to small values (1-3) for optimal search frame rate and new track generation. Setting the number of confirmations to 1 will allow for fastest track generation while setting to a larger value will help reduce spurious track generation.

Active Mission	Default	C-UAS_1	C-UAS_2	COASTAL_C-UAS	DISMOUNT
Default Value	1	—	—	—	3

Attribute	Value
Type	integer
Constraints	$1 \leq x \leq 20$
Nullable	No
Modifiable Mode	Operation and Service

continues on next page

Table 183 – continued from previous page

Attribute	Value
Persistence	Volatile
User Access	SUPER_USER

track_elmax

Tracking field-of-view maximum elevation boundary in degrees. Tracks outside this range will not be updated by the radar and will coast for less than or equal to the Max Out of Bounds Coast Time (track_max_oob_coast_time_s) before being deleted by the tracker. Special constraint: track_elmin < track_elmax

Attribute	Value
Default	50
Type	number
Constraints	$-40 \leq x \leq 50$
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Volatile
User Access	USER (default)

track_elmin

Tracking field-of-view minimum elevation boundary in degrees. Tracks outside this range will not be updated by the radar and will coast for less than or equal to the Max Out of Bounds Coast Time (track_max_oob_coast_time_s) before being deleted by the tracker. Special constraint: track_elmin < track_elmax

Attribute	Value
Default	-40
Type	number
Constraints	$-40 \leq x \leq 50$
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Volatile
User Access	USER (default)

track_history_gate_level

The chi-squared statistic threshold used for associating and merging tracks. Increasing this value will reduce the probability two tracks will form on the same target, while reducing this value will allow tracks to form on separate targets with slightly different positions and velocities.

Attribute	Value
Default	30
Type	number
Constraints	$0 < x \leq 200$
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Volatile
User Access	SUPER_USER

track_max_coast_time_s

This parameter controls the minimum amount of time in which a track will go from a confidence level of 100% to a confidence level of 0% when no measurements are associated to the track and the track is still within the tracking field-of-view. It is recommended to keep this value between 2 and 10 seconds. In addition, it is recommended to always keep this value greater than track_max_confidence_time_s.

Active Mission	Default	C-UAS_1	C-UAS_2	COASTAL_C-UAS	DISMOUNT
Default Value	5	—	—	—	10.0

Attribute	Value
Type	number
Constraints	$0 < x \leq 20$
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Volatile
User Access	SUPER_USER

track_max_confidence_time_s

The amount of time in seconds that a track must be observed, with high probability measurements associated to it, to attain a 100% confidence level. This should typically be several times smaller than track_max_coast_time_s.

Active Mission	Default	C-UAS_1	C-UAS_2	COASTAL_C-UAS	DISMOUNT
Default Value	1	—	—	—	2.0

Attribute	Value
Type	number
Constraints	$0 < x \leq 20$
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Volatile
User Access	SUPER_USER

track_max_oob_coast_time_s

This parameter controls the maximum amount of time a track can be outside the track field-of-view before the track confidence decays to 0. Note that the track and search fields of view can be different and are configured independently

Attribute	Value
Default	1
Type	number
Constraints	$0 < x \leq 20$
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Volatile
User Access	SUPER_USER

track_min_reporting_informed_track_update_count

Sets the minimum number of informed track updates required before a given track is reported to the user. An informed track update is a track update where at least one measurement is associated to the track.

Attribute	Value
Default	0
Type	integer
Constraints	$0 \leq x \leq 1000$
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Volatile
User Access	SUPER_USER

track_report_unconfirmed_tracks

Flag representing whether unconfirmed tracks are reported to the user. NOTE: This setting is superseded by network:tracks_output_report_all_tracks_enable. If network:tracks_output_report_all_tracks_enable = true, unconfirmed tracks will be output to the user even if track_report_unconfirmed_tracks = false.

Attribute	Value
Default	False
Type	boolean
Constraints	[True, False]
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Volatile
User Access	SUPER_USER

track_update_beam_adaptive_pfa_enable

Boolean representing whether adaptive track beam pfa feature is enabled. When this feature is enabled, the system will use a higher CFAR PFA in the region surrounding the track being interrogated. Enabling this feature will increase the probability of detecting targets of interest and thus increase tracking ranges, but will also increase the probability of generating tracks on clutter. It is recommend to only use this feature in low clutter environments.

Attribute	Value
Default	False

continues on next page

Table 194 – continued from previous page

Attribute	Value
Type	boolean
Constraints	[True, False]
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Volatile
User Access	SUPER_USER

track_update_rate_hz

Controls the rate at which each track's state estimate is internally updated. Decreasing this value will increase the latency between when a target is observed and when that information is used to update a track's state estimate. In general, decreasing this value will decrease track accuracy.

Attribute	Value
Default	10
Type	number
Constraints	$1 \leq x \leq 10$
Nullable	No
Modifiable Mode	Client Read Only (FW Modifiable)
Persistence	Volatile
User Access	USER (default)

tx_channel

Radar transmit channel

Attribute	Value
Default	"U8"
Type	enum

continues on next page

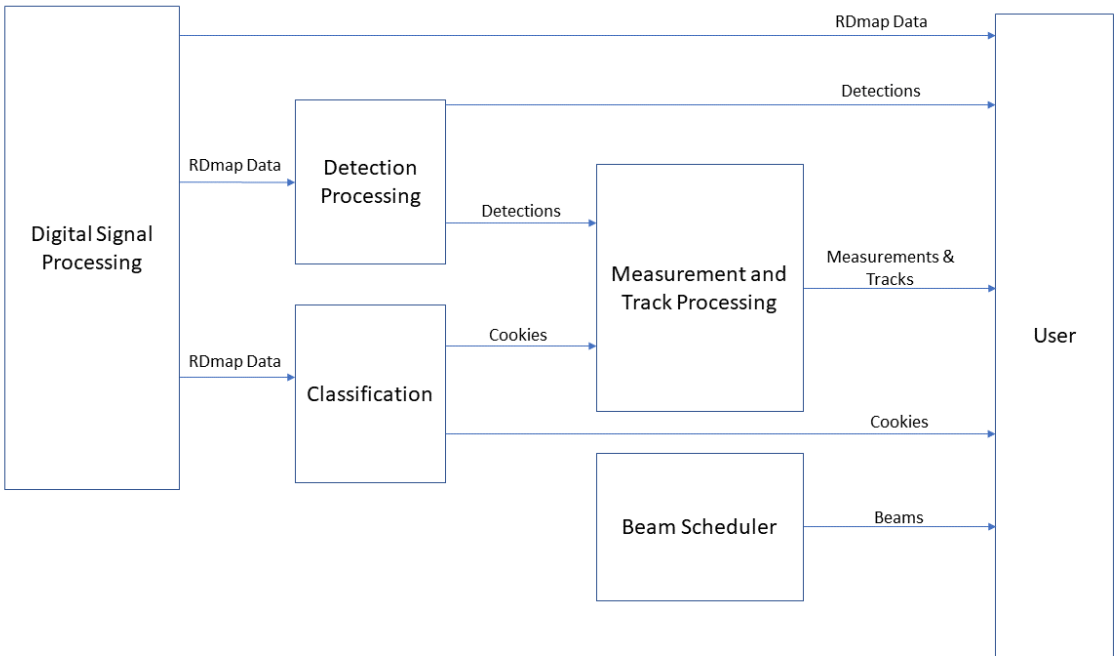
Table 196 – continued from previous page

Attribute	Value
Constraints	• L1 • L2 • L3 • L4 • L5 • U1 • U2 • U3 • U4 • U5 • U6 • U7 • U8 • U9 • U10 • U11 • U12 • U13 • U14 • U15 • U16 • U17
Nullable	No
Modifiable Mode	Operation and Service
Persistence	Sticky
User Access	USER (default)

5 Data

This section documents the format of each data packet type made available to users for capture. Raw radar data in the form of *Rdmaps* are made available on the 10Gb interface. Processed *Tracks*, *Detections* and *Measurements* are transmitted on the 1Gb interface, along with *Beams*, *Status*, and *Kinematics* data.

The following flow-chart is a brief overview of radar data packet processing:



5.1 Variable Packet Size

Several data types may vary in individual packet size. These packets contain a *fixed* section that always appears once at the start of a packet, and 0 or more *variable* sections appended at the end of the packet based on the amount of data in the pipeline.

For example, a *Detections* packet has a *fixed* section size of 64 bytes (header), and a *variable* section size of 184 bytes (detections). The number of detections in the packet is reported by the *n_detect* field in the header. So the total packet size is as the following table shows:

n_detect	Total *fixed* size	Total *variable* size	Total packet size
0	64B	0B	64B
1	64B	184B	248B
2	64B	368B	432B
100	64B	18400B	18464B

The following packet types have variable packet sizes:

- *Detections*
- *Measurements*

5.2 1Gb Interface

This section describes the following output packet schemas:

- *Tracks*
- *Representative*
- *Status*
- *Kinematics*
- *Detections*
- *Beams*
- *Measurements*
- *Cookies*

Tracks

Contains data about active tracks, including state and class estimates for reported targets. Track packets are transmitted at a configurable rate per the *tracks_output_rate_hz* parameter. In a single track output period, one NULL track packet and one standard track packet per active track are generated. NULL track packets are packets with header information only that are indicated with an 'id' field equal to '0'. These packets may serve as separators between track-output periods, as they are always the first packet sent per period.

Packet data is in little-endian byte order.

Track packet, Gen3

Table 197: fixed

Byte	C Type	Name	Unit	Description
0	char[8]	packet_sync	ASCII	Denotes start of new packet **Constraints**: <tracks>

continues on next page

Table 197 – continued from previous page

Byte	C Type	Name	Unit	Description
8	uint32_t	n_bytes	bytes	Total number of bytes in this packet **Constraints**: Standard: 504 Extended: 728
12	uint8_t	version_major	Version #	Major version number of the Gen-3 track packet
13	uint8_t	version_minor	Version #	Minor version number of the Gen-3 track packet
14	uint8_t	version_patch	Version #	Patch version number of the Gen-3 track packet
15	uint8_t[1]	reserved_00	N/A	Echodyne reserved field
16	uint32_t	radar_id	ID #	Unique numeric ID for this radar. Built of the lower 4 bytes of the system's 1G MAC address in little endian byte order.
20	uint8_t	packet_type	N/A	Indicates whether track packets are in standard (value=0) or extended format (value=1) **Constraints**: $[0, 1]$
21	uint8_t	state	N/A	Track state **Constraints**: $[0, 1, 2, 3, 4]$ *state value descriptions*
22	uint8_t[6]	reserved_01	N/A	Echodyne reserved field
28	uint32_t	lifetime	track updates	The number of updates the track has been active for
32	float	confidence_level	%	0-100 value representing how likely the track is to be accurate **Constraints**: $0 \leq C \leq 100$
36	uint32_t	informed_track_update_count	track updates	The number of track updates where measurements were associated to this track
40	uint8_t[8]	reserved_02	N/A	Echodyne reserved field

continues on next page

Table 197 – continued from previous page

Byte	C Type	Name	Unit	Description
48	uint64_t	id	ID #	Locally incrementing track identifier. This ID will be unique for a single radar over the time period since the radar was last power cycled. Please use track_UUID if you need a universally unique track ID. A track ID of 0 is reserved for NULL track packets.
56	uint8_t[16]	track_UUID	ID #	Universally unique ID for this track. Track UUIDs are created using a version 4 UUID generator. A track_UUID of all zeros is reserved for NULL track packets.
72	uint8_t[16]	handoff_UUID	ID #	Universally unique ID of the external track handoff that was last associated to this track. This will be all zeros if no external handoff has been associated to this track.
88	uint8_t[16]	track_merge_UUID	ID #	Universally unique ID of the track that this track merged into. This field is only meaningful for obituary track updates where the track died due to a track merge. In all other cases, this value will be all zeros.
104	float[3]	xyz_pos_est	m	X/Y/Z position estimate in Radar coordinate system *xyz_est mapping*
116	float[3]	xyz_vel_est	m/s	X/Y/Z velocity estimate in Radar coordinate system *xyz_velocity_estimate mapping*
128	double[3]	ecef_pos_est	m	X/Y/Z position estimate in ECEF
152	float[3]	ecef_vel_est	m/s	X/Y/Z velocity estimate in ECEF
164	float[3]	enu_pos_est	m	East/North/Up position estimate in radar centered local ENU frame *enu_est mapping*

continues on next page

Table 197 – continued from previous page

Byte	C Type	Name	Unit	Description
176	float[3]	enu_vel_est	m/s	East/North/Up velocity estimate in radar centered local ENU frame *vel_enu_est mapping*
188	float	rcs_est	dBsm	Radar Cross-Section estimate in dBsm **Constraints**: $-130 < RCS \leq 100$
192	float	rcs_est_std	dBsm	Standard deviation of Radar Cross-Section estimate in dBsm **Constraints**: $0 \leq RCSstd$
196	uint8_t	track_formation_source	N/A	Describes how the track formed **Constraints**: $[0, 1]$ *state value descriptions*
197	uint8_t	track_cause_of_death	N/A	Describes how the track died. This information is only meaningful for obituary track updates. **Constraints**: $[0, 1, 2, 3, 4]$ *state value descriptions*
198	uint8_t	track_is_focused	N/A	Integer boolean indicating whether the radar is focusing on this track as controlled by the track_focus_start command **Constraints**: $[0, 1]$
199	uint8_t[1]	reserved_03	N/A	Echodyne reserved field
200	int64_t	last_update_time	ns	Last update time in Unix time
208	int64_t	last_assoc_time	ns	Last associated data time in Unix time
216	int64_t	acquired_time	ns	Time initially acquired in Unix time
224	float	agl_est	m	Height above ground level
228	float	prob_aircraft	probability	The predicted probability the track is of class *aircraft* **Constraints**: $0 \leq prob \leq 1$

continues on next page

Table 197 – continued from previous page

Byte	C Type	Name	Unit	Description
232	float	prob_bird	probability	The predicted probability the track is of class *bird* **Constraints**: $0 \leq prob \leq 1$
236	float	prob_clutter	probability	The predicted probability the track is of class *clutter* **Constraints**: $0 \leq prob \leq 1$
240	float	prob_human	probability	The predicted probability the track is of class *human* **Constraints**: $0 \leq prob \leq 1$
244	float	prob_uav_fixedwing	probability	The predicted probability the track is of class *uav fixedwing* **Constraints**: $0 \leq prob \leq 1$
248	float	prob_uav_multirotor	probability	The predicted probability the track is of class *uav multirotor* **Constraints**: $0 \leq prob \leq 1$
252	float	prob_vehicle	probability	The predicted probability the track is of class *vehicle* **Constraints**: $0 \leq prob \leq 1$
256	uint8_t[32]	reserved_04	N/A	Echodyne reserved field
288	float[21]	ecef_state_covariance	covariance	The upper right triangular matrix representation of the *a posteriori* estimate covariance matrix. This represents the track state estimate's ECEF position and velocity uncertainty. *Upper right triangular matrix representation of the a posteriori estimate covariance matrix*
372	uint8_t[132]	reserved_05	N/A	Echodyne reserved field

Extended Track packet

Table 198: fixed

Byte	C Type	Name	Unit	Description
0	uint8_t	n_outstanding_track_beams	beams	Number of beams the track has requested but not received data for
1	uint8_t	n_outstanding_clf_beams	beams	Number of beams the classifier has requested but not received data for
2	uint8_t	n_assoc_meas_ids	#	Number of valid ID's in the assoc_meas_ids field
3	uint8_t	n_assoc_cookie_ids	#	Number of valid ID's in the assoc_cookie_ids field
4	float	assoc_meas_mean_adjusted_rcs	dBsm	Mean RCS of measurements associated to this track during the current track update period accounting for any adjustments made to the measurement in the measurement to track association process. If SQL and CLUSTER measurements were associated to the track, only the RCS from SQL measurements is used to compute the mean RCS. **Constraints**: $-130 < RCS \leq 100$, NaN if disabled
8	float[6]	assoc_meas_chi2	chi^2	χ^2 metric of Associated Measurements
32	uint64_t[6]	assoc_meas_ids	ID #	Associated measurement IDs. Only the first 'n_assoc_meas_ids' fields are valid
80	uint64_t[2]	outstanding_clf_beams_ids	ID #	IDs of outstanding classifier beams. Only the first 'n_outstanding_clf_beams_ids' fields are valid.
96	int64_t	last_clf_beam_time	ns	Last classifier beam time in Unix time

continues on next page

Table 198 – continued from previous page

Byte	C Type	Name	Unit	Description
104	uint64_t[4]	outstanding_track_beams_ids	ID #	IDs of outstanding track beams. Only the first 'n_outstanding_track_beams_ids' fields are valid.
136	int64_t	last_track_beam_time	ns	Last track beam time in Unix time
144	uint64_t[2]	assoc_cookie_ids	ID #	Associated cookie IDs. Only the first 'n_assoc_cookie_ids' fields are valid.
160	uint8_t[64]	reserved_06	N/A	Echodyne reserved field

Tracks Packet Field Objects

state value descriptions

Value	State
0	INACTIVE
1	UNCONFIRMED
2	CONFIRMED
3	AMBIGUOUS
4	HANDOFF

xyz_est mapping

Index	Coordinate
0	X
1	Y
2	Z

xyz_velocity_estimate mapping

Index	Coordinate
0	X
1	Y
2	Z

enu_est mapping

Index	Direction
0	East
1	North
2	Up

vel_enu_est mapping

Index	Direction
0	East
1	North
2	Up

state value descriptions

Value	State
0	INTERNAL
1	HANDOFF

state value descriptions

Value	State
0	NA
1	KILLED_BY_MERGE
2	KILLED_BY_COAST
3	INVALID_STATE
4	TRACKER_STOPPED

Upper right triangular matrix representation of the *a posteriori* estimate covariance matrix

Upper right triangular matrix representation of the *a posteriori* estimate covariance matrix, transmitted in row-major order. The values in the provided table map to the index in the state_covariance array. Positions are measured in meters, velocities are measured in meters per second.

	X Pos. (0)	X Vel. (1)	Y Pos. (2)	Y Vel. (3)	Z Pos. (4)	Z Vel. (5)
X Pos. (0)	0	1	2	3	4	5
X Vel. (1)	…	6	7	8	9	10
Y Pos. (2)	…	…	11	12	13	14
Y Vel. (3)	…	…	…	15	16	17
Z Pos. (4)	…	…	…	…	18	19
Z Vel. (5)	…	…	…	…	…	20

Representative

A representative track comes from a track group and summarizes that group. A track group is defined as a set of tracks that all represent the same target. Track groups are formed by associating tracks from multiple radars that are on the same target and is performed by some central processing routine. A representative track contains all of the same core information as a radar track such as target state and class estimates. Representative track updates are emitted asynchronously at the rate the representative state estimate is updated. This should correspond to the rate at which the central processing unit receives track updates from the various radars it is connected to.

Packet data is in little-endian byte order.

Representative Track packet, Gen3

Table 207: fixed

Byte	C Type	Name	Unit	Description
0	char[8]	packet_sync	ASCII	Denotes start of new packet **Constraints**: <reptrk>
8	uint32_t	n_bytes	bytes	Total number of bytes in this packet **Constraints**: Standard: 504 Extended: variable
12	uint8_t	version_major	Version #	Major version number of the Gen-3 representative track packet
13	uint8_t	version_minor	Version #	Minor version number of the Gen-3 representative track packet
14	uint8_t	version_patch	Version #	Patch version number of the Gen-3 representative track packet
15	uint8_t[5]	reserved_00	N/A	Echodyne reserved field
20	uint8_t	packet_type	N/A	Indicates whether representative track packets are in standard (value=0) or extended format (value=1) **Constraints**: $[0, 1]$
21	uint8_t	state	N/A	Representative Track state **Constraints**: $[0, 1, 2, 3, 4]$ *state value descriptions*
22	uint8_t[6]	reserved_01	N/A	Echodyne reserved field
28	uint32_t	lifetime	representative track updates	The number of updates the representative track has been active for
32	float	confidence_level	%	0-100 value representing how likely the representative track is to be accurate **Constraints**: $0 \leq C \leq 100$

continues on next page

Table 207 – continued from previous page

Byte	C Type	Name	Unit	Description
36	uint32_t	informed_track_update_count	representative track updates	The number of representative track updates where measurements were associated to a radar track used to update this representative track
40	uint8_t[8]	reserved_02	N/A	Echodyne reserved field
48	uint64_t	id	ID #	Locally incrementing representative track identifier. This ID will be unique for a single instance of EchoWare over the time period since track handoff and grouping was last initialized. Please use track_UUID if you need a universally unique representative track ID. A representative track ID of 0 is reserved for NULL representative track packets.
56	uint8_t[16]	track_UUID	ID #	Universally unique ID for this representative track. Representative track UUIDs are created using a version 4 UUID generator. The representative track track_UUID is expected to be unique over all representative tracks and radar tracks. The radar track track_UUID's used to form this representative track can be found in the extended portion of this packet. A track_UUID of all zeros is reserved for NULL representative track packets.
72	uint8_t[16]	reserved_03	N/A	Echodyne reserved field
88	uint8_t[16]	track_merge_UUID	ID #	Universally unique ID of the representative track that this representative track merged into. This field is only meaningful for obituary representative track updates where the representative track died due to a representative track merge. In all other cases, this value will be all zeros.

continues on next page

Table 207 – continued from previous page

Byte	C Type	Name	Unit	Description
104	uint8_t[24]	reserved_04	N/A	Echodyne reserved field
128	double[3]	ecef_pos_est	m	X/Y/Z position estimate in ECEF
152	float[3]	ecef_vel_est	m/s	X/Y/Z velocity estimate in ECEF
164	uint8_t[24]	reserved_05	N/A	Echodyne reserved field
188	float	rcs_est	dBsm	Radar Cross-Section estimate in dBsm **Constraints**: $-130 < RCS \leq 100$
192	float	rcs_est_std	dBsm	Standard deviation of Radar Cross-Section estimate in dBsm **Constraints**: $0 \leq RCSstd$
196	uint8_t[1]	reserved_06	N/A	Echodyne reserved field
197	uint8_t	track_cause_of_death	N/A	Describes how the representative track died. This information is only meaningful for obituary representative track updates. **Constraints**: $[0, 1, 2, 3, 4]$ *state value descriptions*
198	uint8_t	track_is_focused	N/A	Integer boolean indicating whether a radar is focusing on this representative track as controlled by the track_focus_start command **Constraints**: $[0, 1]$
199	uint8_t[1]	reserved_07	N/A	Echodyne reserved field
200	int64_t	last_update_time	ns	Last update time in Unix time
208	int64_t	last_assoc_time	ns	Last associated data time in Unix time
216	int64_t	acquired_time	ns	Time initially acquired in Unix time
224	float	agl_est	m	Height above ground level
228	float	prob_aircraft	probability	The predicted probability the track is of class *aircraft* **Constraints**: $0 \leq prob \leq 1$

continues on next page

Table 207 – continued from previous page

Byte	C Type	Name	Unit	Description
232	float	prob_bird	probability	The predicted probability the track is of class *bird* **Constraints**: $0 \leq prob \leq 1$
236	float	prob_clutter	probability	The predicted probability the track is of class *clutter* **Constraints**: $0 \leq prob \leq 1$
240	float	prob_human	probability	The predicted probability the track is of class *human* **Constraints**: $0 \leq prob \leq 1$
244	float	prob_uav_fixedwing	probability	The predicted probability the track is of class *uav fixedwing* **Constraints**: $0 \leq prob \leq 1$
248	float	prob_uav_multirotor	probability	The predicted probability the track is of class *uav multirotor* **Constraints**: $0 \leq prob \leq 1$
252	float	prob_vehicle	probability	The predicted probability the track is of class *vehicle* **Constraints**: $0 \leq prob \leq 1$
256	uint8_t[32]	reserved_08	N/A	Echodyne reserved field
288	float[21]	ecef_state_covariance	covariance	The upper right triangular matrix representation of the *a posteriori* estimate covariance matrix. This represents the representative track state estimate's ECEF position and velocity uncertainty. *Upper right triangular matrix representation of the a posteriori estimate covariance matrix*
372	uint8_t[132]	reserved_09	N/A	Echodyne reserved field

Extended Representative Track packet

Table 208: fixed

Byte	C Type	Name	Unit	Description
0	uint8_t[16]	nominee_UUID	ID #	UUID of the currently-selected nominee radar track for this group
16	uint32_t	n_radar_tracks_in_group	#	Number of radar tracks in the track group used to generate this representative track.
20	uint8_t[100]	reserved_10	N/A	Echodyne reserved field

Table 209: variable

Byte	C Type	Name	Unit	Description
0	uint8_t[16]	radar_track_UUID	ID #	UUID of a radar track that is currently a member of the track group used to form this representative track update.
16	uint32_t	radar_track_lifetime	radar track updates	The current radar track lifetime for the radar track specified by radar_track_UUID. This helps map to the specific radar track update used when generating this representative track update.

Representative Track Packet Field Objects

state value descriptions

Value	State
0	INACTIVE
1	UNCONFIRMED
2	CONFIRMED
3	AMBIGUOUS
4	HANDOFF

state value descriptions

Value	State
0	NA
1	KILLED_BY_MERGE
2	KILLED_BY_COAST
3	INVALID_STATE
4	TRACKER_STOPPED

Upper right triangular matrix representation of the *a posteriori* estimate covariance matrix

Upper right triangular matrix representation of the *a posteriori* estimate covariance matrix, transmitted in row-major order. The values in the provided table map to the index in the state_covariance array. Positions are measured in meters, velocities are measured in meters per second.

	X Pos. (0)	X Vel. (1)	Y Pos. (2)	Y Vel. (3)	Z Pos. (4)	Z Vel. (5)
X Pos. (0)	0	1	2	3	4	5
X Vel. (1)	…	6	7	8	9	10
Y Pos. (2)	…	…	11	12	13	14
Y Vel. (3)	…	…	…	15	16	17
Z Pos. (4)	…	…	…	…	18	19
Z Vel. (5)	…	…	…	…	…	20

Status

Contains data about current system state. Status packets are sent at a user-configurable nominal rate, which defaults to 1 Hz. On special events, such as time source status, PLL Lock status, safety-interlock state, system mode, SBIT status or CBIT status, a status packet is sent immediately. Status ports are enabled as long as the system is up and running, regardless to whether it is actively radaring or whether it has entered fault state (except when there is a fault in the 1G network interface). Packet data is in little-endian byte order.

Status packet

Table 213: fixed

Byte	C Type	Name	Unit	Description
0	char[8]	packet_sync	ASCII	Denotes start of new packet **Constraints**: <status>
8	uint32_t	n_bytes	bytes	Total number of bytes **Constraints**: 1320
12	uint8_t	version_major	Version #	Major version number of the Gen-3 status packet **Constraints**: 1
13	uint8_t	version_minor	Version #	Minor version number of the Gen-3 status packet **Constraints**: 0
14	uint8_t	version_patch	Version #	Patch version number of the Gen-3 status packet **Constraints**: 0
15	char[64]	unit_serial	ASCII	Unit serial number
79	uint8_t	system_health_status	N/A	The system_health_status field summarizes the results of the Startup Built-In Test (SBIT) and Continuous Built-In Test (CBIT) processes. These tests monitor the health of the system to detect and report any issues early *system_health_status descriptions*
80	char[64]	system_mode	ASCII	Current mode of operation *system_mode value descriptions*
144	char[128]	fault_reason	ASCII	Fault reason, when mode of operation is fault
272	uint32_t	transmitter_output_power_control	N/A	Status of transmitter output power control *transmitter_output_power_control value descriptions*
276	uint32_t	radar_uptime	s	Time the radar has spent powered on since it was manufactured, in seconds

continues on next page

Table 213 – continued from previous page

Byte	C Type	Name	Unit	Description
280	uint32_t	net_state_1g	N/A	Network 1Gb interface state enumeration *net_state_1g value descriptions*
284	uint32_t	link_status_10g	N/A	link status of the 10G interface. 0 -> link is down, !0 -> link is up
288	int64_t	system_time	ns	Timepoint corresponding to UTC time. Suitable for timestamping events, but should not be used to calculate precise durations between events
296	uint32_t	active_time_source	N/A	Active time source enumeration *active_time_source value descriptions*
300	uint32_t	available_time_sources	N/A	Available time sources - bit field *time_sources bits*
304	float	search_scan_rate	Hz	Number of times the radar scans a FoV frame per second
308	uint32_t	kinematics_sensor_agreement	N/A	Enumeration describing level of agreement between onboard INS solution and user-set kinematics. A user can check get_ins_data and get_radar_kinematics, or view the radar logs, for numerical details. The INS solution is expected to be invalid when moving *kinematics_sensor_agreement value descriptions*
312	uint8_t[1008	reserved_00	N/A	Echodyne reserved field

Status Packet Field Objects

system_health_status descriptions

Value	State	Description
0	BIT_FAULT	The system has failed one or more tests during SBIT or CBIT. Immediate attention is required
1	BIT_WARNING	The system has passed the tests, but there are warnings or potential issues identified during eit
2	BIT_OK	The system has successfully passed both SBIT and CBIT without any errors or warnings. The

system_mode value descriptions

Mode	Description
Resetting	Awaiting system reset
Reset	System reset complete
Initializing	System is initializing
Operation	System is in Operation mode
Operation:IDLE	System is idle
Operation:TX_ACTIVE	Radar transmitter is firing
Calibration:TX_ACTIVE	Radar transmitter is firing, calibration mode
Service	System is in Service mode
Fault	System is in Fault mode
Standby	System is in Standby/Wait mode

transmitter_output_power_control value descriptions

Value	Status
0	UNKNOWN
1	TX_OFF
2	TX_LOW_POWER
3	TX_FULL_POWER

net_state_1g value descriptions

Value	State
0	STATUS_UNKNOWN
1	HALF_DUPLEX_10M
2	HALF_DUPLEX_100M
3	HALF_DUPLEX_1G
4	FULL_DUPLEX_10M
5	FULL_DUPLEX_100M
6	FULL_DUPLEX_1G

active_time_source value descriptions

Value	Time Source
0	NTP1
1	NTP2
2	NTP3
3	N/A - Reserved
4	NMEA
5	PPS
6	None

time_sources bits

Bit	Time Source
31-6	N/A - Reserved
5	PPS
4	NMEA
3	N/A - Reserved
2	NTP3
1	NTP2
0	NTP1

kinematics_sensor_agreement value descriptions

Value	Sensor Agreement
0	Green: Agreement
1	Yellow: some disagreement between INS solution and user-set kinematics
2	Red: large amount of disagreement between INS solution and user-set kinematics
4	N/A: GPS solution, INS solution, or user-set kinematics unavailable

Kinematics

Contains radar kinematics data for rotation and position. Packet data is in little-endian byte order.

Kinematics packet

Table 221: fixed

Byte	C Type	Name	Unit	Description
0	char[8]	packet_sync	ASCII	Denotes start of new packet **Constraints**: <kinems>
8	uint32_t	n_bytes	bytes	Total number of bytes **Constraints**: 128
12	uint8_t	version_major	Version #	Major version number of the Gen-3 kinematics packet
13	uint8_t	version_minor	Version #	Minor version number of the Gen-3 kinematics packet
14	uint8_t	version_patch	Version #	Patch version number of the Gen-3 kinematics packet
15	uint8_t[1]	reserved_00	N/A	Echodyne reserved field
16	uint64_t	update_id	ID #	Unique id of kinematics update
24	int64_t	time	ns	Unix time of kinematics measurement
32	float[4]	radar_ecef_quaternion	quaternion	Radar orientation quaternion as qx, qy, qz, qw
48	double[3]	radar_ecef_position	m	Radar ECEF position
72	float[3]	radar_ecef_angular_velocity	rad/s	Radar angular velocity vector in ECEF frame
84	float[3]	radar_ecef_velocity	m/s	Radar ECEF velocity
96	uint8_t	kinematics_source_internal_ins	N/A	Integer boolean indicating the usage of an external reference, such as an INS **Constraints**: [0, 1]

continues on next page

Table 221 – continued from previous page

Byte	C Type	Name	Unit	Description
97	uint8_t	timestamp_source_external	N/A	Integer boolean indicating if the timestamp was set from an external source, such as a GPS or user-provided Unix timestamp, instead of the internal system clock **Constraints**: [0, 1]
98	uint8_t[78]	reserved_01	N/A	Echodyne reserved field

Detections

Contains data related to Constant-False-Alarm-Rate (CFAR) detections for a given beam. Detections combine raw radar data collected in Range-Doppler maps with system properties to produce derived parameters such as target range, doppler, and RCS. Detection data is reported for each beam that has CFAR detection processing enabled. Packet data is in little-endian byte order.

Detections packet

Table 222: fixed

Byte	C Type	Name	Unit	Description
0	char[8]	packet_sync	ASCII	Denotes start of new packet **Constraints**: <detect>
8	uint32_t	n_bytes	bytes	Total number of bytes in this packet **Constraints**: 64 + 184 * 'n_detect'
12	uint8_t	version_major	Version #	Major version number of the Gen-3 detections packet
13	uint8_t	version_minor	Version #	Minor version number of the Gen-3 detections packet
14	uint8_t	version_patch	Version #	Patch version number of the Gen-3 detections packet
15	uint8_t[1]	reserved_00	N/A	Echodyne reserved field
16	uint64_t	beam_id	ID #	Generating beam's ID
24	int64_t	beam_time	ns	Generating beam's trigger time in Unix time

continues on next page

Table 222 – continued from previous page

Byte	C Type	Name	Unit	Description
32	uint32_t	n_detect	detections	Number of detections in this packet **Constraints**: $0 \leq x \leq 2800$
36	uint32_t	beam_type	N/A	Enumeration: type of Beam
40	float	beam_gain	dBi	Peak two-way gain of the beam in dBi used to generate the detection
44	uint8_t[20]	reserved_01	N/A	Echodyne reserved field

Table 223: variable

Byte	C Type	Name	Unit	Description
0	float[2]	uv_est	U/V coordinates	Estimate in U/V coordinates
8	float[2]	uv_variance	variance	Variance of U/V estimate
16	float[2]	rd_est	[m, m/s]	Estimate in Range/Doppler coordinates
24	float[2]	rd_variance	[m^2, (m/s)^2]	Variance of Range/Doppler estimate
32	double[3]	ecef_est	m	X/Y/Z position estimate in ECEF
56	float[3]	enu_est	m	East/North/Up position estimate in ENU
68	float	power	dB ADC^2	Magnitude squared of complex sum field
72	float	snr_dB	dB	Signal-to-noise ratio
76	float[2]	rd_scalloping_loss	dB	Estimated scalloping loss from windowing
84	float	rcs_est	dBsm	Radar Cross-Section estimate in dBsm **Constraints**: $-130 < RCS \leq 100$
88	uint16_t[2]	index_rdmap	N/A	Row (range), Column (Doppler) indices in RDMap
92	uint8_t[44]	reserved_02	N/A	Echodyne reserved field
136	int32_t	cluster_size	detections	size of the detection cluster. -1 indicates clustering was not performed

continues on next page

Table 223 – continued from previous page

Byte	C Type	Name	Unit	Description
140	uint8_t	monopulse_used	N/A	Flag: If true, Monopulse was used to improve the U estimate **Constraints**: 0, 1
141	uint8_t	interpolation_used	N/A	Flag: If true, Doppler interpolation was used to improve the Doppler estimate **Constraints**: 0, 1
142	uint16_t	reject_mask	bit-field	Bitmap indicating whether this detection has been filtered by one of the Detection Rejection Masks
144	uint64_t	id	ID #	Unique detection ID
152	uint8_t[32]	reserved_03	N/A	Echodyne reserved field

Beams

A beam object contains the set of instructions used to configure the radar to interrogate the environment over a single coherent processing interval (CPI). This information includes things such as interrogation angle and waveform. Packet data is in little-endian byte order.

Beams packet, Gen3

Table 224: fixed

Byte	C Type	Name	Unit	Description
0	char[8]	packet_sync	ASCII	Denotes start of new packet **Constraints**: <beams!>
8	uint32_t	n_bytes	bytes	Total number of bytes in this packet **Constraints**: 296
12	uint8_t	version_major	Version #	Major version number of the Gen-3 beams packet
13	uint8_t	version_minor	Version #	Minor version number of the Gen-3 beams packet

continues on next page

Table 224 – continued from previous page

Byte	C Type	Name	Unit	Description
14	uint8_t	version_patch	Version #	Patch version number of the Gen-3 beams packet
15	uint8_t[5]	reserved_00	N/A	Echodyne reserved field
20	float[2]	uv	U/V coordinates	The beam's U/V values in sine-space
28	uint8_t[4]	reserved_01	N/A	Echodyne reserved field
32	uint32_t	type	N/A	Enumeration: type of Beam **Constraints**: $0 \leq x \leq 14$ *type value descriptions*
36	uint32_t	request_source	N/A	Enumeration: Beam requester **Constraints**: $0 \leq x \leq 3$ *request_source value descriptions*
40	uint32_t	n_pulse	pulses	Number of pulses **Constraints**: $1 \leq x \leq 4096$
44	uint32_t	t_pulse	ns	Time duration of pulse
48	uint32_t	t_sampling_delay	ns	Time duration from beginning of pulse to when receiver begins sampling
52	uint32_t	t_pri	ns	Time duration of Pulse Repetition Interval
56	uint32_t	f0	kHz	Center frequency of transmitted signal
60	uint32_t	BW	Hz	Bandwidth of transmitted signal
64	uint32_t	decimation_factor	N/A	Integer factor that the ADC sample rate is decimated by
68	float[2]	rd_subsection_start	[m, m/s]	Range (m) and doppler (m/s) of zeroth range/Doppler bin of returned RDMap subsection
76	float[2]	rd_sample_resolution	[m, m/s]`	Range (m) and doppler (m/s) width of each range/Doppler bin
84	uint16_t[2]	n_subsection	RD bins	Number of Range/Doppler bins in returned RDMap subsection
88	uint8_t	fpga_dets_enabled	N/A	Flag: If true, FPGA detections are enabled
89	uint8_t[3]	reserved_02	N/A	Echodyne reserved field
92	uint8_t[4]	channel_enabled	N/A	Flag: If true, FPGA is outputting data for channels 0, 1, 2, 3

continues on next page

Table 224 – continued from previous page

Byte	C Type	Name	Unit	Description
96	uint32_t[4]	channel_source	N/A	Enumeration: FPGA data source for channels 0, 1, 2, 3
112	uint32_t[4]	channel_format	N/A	Enumeration: FPGA data format for channels 0, 1, 2, 3
128	uint16_t	cfar_n_train	training cells	Number of training cells used in CFAR processing
130	uint16_t	cfar_n_guard	guard cells	Number of guard cells used in CFAR processing
132	float	cfar_pfa	probability	Probability of false alarm rate used in CFAR processing
136	float[2]	target_rd_est	[m, m/s]	Estimated Range/Doppler used by high fidelity classifier for target of interest identification
144	uint64_t	id	ID #	Unique ID of beam
152	uint64_t	sequence_num	ID #	Sequence number, incrementing by one from 0 after entering initialized state
160	int64_t	request_time	ns	Beam's request time in Unix time
168	uint8_t[128]	reserved_03	N/A	Echodyne reserved field

Beams Packet Field Objects

type value descriptions

Value	Beam Type
0	BACKGROUND_SEARCH
1	HORIZON_SEARCH
2	UNCONFIRMED_TRACK
3	CONFIRMED_TRACK
4	CLASSIFICATION
5	USER
6	DEBUG
7	ENVIRONMENT_MONITOR
8	MONOPULSE_CALIBRATION
9	RFFE_CALIBRATION
10	RFFEAF_CALIBRATION
11	USER_CALIBRATION
12	SEARCH

continues on next page

Table 225 – continued from previous page

Value	Beam Type
13	GROUND_AREA_TRACK
14	SPECTRUM_TEST

request_source value descriptions

Value	Request Source
0	BEAMSCHEDULER
1	TRACKER
2	USER
3	CALIBRATOR

Measurements

Contains data about measurements created during the current track update period. Measurements are refined detections from inter and intrabeam clusters, as well as sequential lobing beams. The resulting measurement is a single estimate of target state. Measurements are transmitted at the same rate as tracks (at the user specified rate between [1.0, 10.0] times per second). Packet data is in little-endian byte order.

Measurements packet, Gen3

Table 227: fixed

Byte	C Type	Name	Unit	Description
0	char[8]	packet_sync	ASCII	Denotes start of new packet **Constraints**: <measur>
8	uint32_t	n_bytes	bytes	Total number of bytes **Constraints**: 160 + 8 * 'n_detection_id'
12	uint8_t	version_major	Version #	Major version number of the Gen-3 measurements packet
13	uint8_t	version_minor	Version #	Minor version number of the Gen-3 measurements packet
14	uint8_t	version_patch	Version #	Patch version number of the Gen-3 measurements packet
15	uint8_t[5]	reserved_00	N/A	Echodyne reserved field

continues on next page

Table 227 – continued from previous page

Byte	C Type	Name	Unit	Description
20	uint32_t	meas_type	N/A	Indicates whether measurement originates from cluster (value=0) or SQL (value=1) **Constraints**: 0, 1
24	int64_t	time	ns	Mean Unix time of detections used to generate the measurement
32	uint64_t	id	ID #	Unique ID of measurement
40	uint8_t[8]	reserved_01	N/A	Echodyne reserved field
48	float[2]	uv_est	U/V coordinates	Estimate in U/V coordinates
56	float[2]	uv_variance	variance	Variance of U/V estimate
64	float[2]	rd_est	[m, m/s]	Estimate in Range/Doppler coordinates
72	float[2]	rd_variance	[m^2, (m/s)^2]	Variance of Range/Doppler estimate
80	float	rcs_est	dBsm	Radar Cross-Section estimate in dBsm **Constraints**: $-130 < RCS \leq 100$
84	float	max_snr	dB	Maximum SNR of contributing detections
88	float	MUV	m/s	Maximum unambiguous velocity of contributing beams
92	float	MUR	m	Maximum unambiguous range of contributing beams
96	double[3]	ecef_est	m	X/Y/Z position estimate in ECEF
120	float[3]	enu_est	m	Positional estimate in ENU
132	uint32_t	n_detection_id	#	Number of `detection_ids` in this measurement **Constraints**: $1 \leq x \leq 512$
136	uint16_t	reject_mask	N/A	Bitmap indicating whether this measurement has been filtered by one of the Measurement Rejection Masks
138	uint8_t[22]	reserved_02	N/A	Echodyne reserved field

Table 228: variable

Byte	C Type	Name	Unit	Description
0	uint64_t	detection_ids	N/A	Detection IDs used in measurement

Cookies

Cookies are a combination of the typical target state information one expects to find in a detection and target class information. The beams used to generate Cookies generally have a large processing gain with high doppler resolution. A set of classifiers is then run on RDmaps and CFAR detections produced by these beams in order to produce an accurate target state and class estimate. Packet data is in little-endian byte order.

Cookies packet, Gen3

Table 229: fixed

Byte	C Type	Name	Unit	Description
0	char[8]	packet_sync	ASCII	Denotes start of new packet **Constraints**: <cookie>
8	uint32_t	n_bytes	bytes	Total number of bytes
12	uint8_t	version_major	Version #	Major version number of the Gen-3 cookies packet **Constraints**: 1
13	uint8_t	version_minor	Version #	Minor version number of the Gen-3 cookies packet **Constraints**: 0
14	uint8_t	version_patch	Version #	Patch version number of the Gen-3 cookies packet **Constraints**: 0
15	uint8_t[1]	reserved_00	N/A	Echodyne reserved field
16	uint64_t	cookie_id	ID #	Unique ID of cookie
24	uint64_t	beam_id	ID #	Generating beam's ID
32	uint32_t	beam_type	N/A	Enumeration: type of Beam **Constraints**: $0 \leq x \leq 14$ *beam_type value descriptions*

continues on next page

Table 229 – continued from previous page

Byte	C Type	Name	Unit	Description
36	float	beam_gain	dBi	Peak two-way gain of the beam in dBi used to generate the detection
40	uint64_t	beam_time	ns	Generating beam's trigger time in Unix time
48	float[2]	target_rd	[m, m/s]	Estimated target range and doppler **Constraints**: $range > 0$
56	float[2]	target_rd_variance	[m^2, (m/s)^2]	Variances of estimated target range and doppler
64	float	target_rcs	dBsm	Estimated target RCS
68	float	target_rcs_variance	(dBsm)^2	Variance of estimated target RCS
72	uint64_t	kinematics_update_id	N/A	Reference kinematics update ID associated with the cookie
80	float[2]	det_rd	[m, m/s]	Detection range and doppler **Constraints**: $range > 0$
88	float[2]	det_rd_variance	[m^2, (m/s)^2]	Variances of detection range and doppler
96	float	det_rcs_est	dBsm	Detection RCS
100	float	quality	N/A	Detection quality
104	uint16_t[2]	rdmap_index	Row (range), Column (Doppler) indices in RDMap	Detection indices in range-doppler map
108	uint8_t[4]	reserved_01	N/A	Echodyne reserved field
112	float	HORD_prob_propeller	N/A	Probability of propeller present from HORD algorithm **Constraints**: $0 \leq p \leq 1$
116	float	HORD_propeller_flash_rate	Hz	Estimated propeller flash rate in Hz from HORD algorithm
120	uint8_t[424]	reserved_02	N/A	Echodyne reserved field
544	uint32_t	r0Subsection	N/A	Range bin index of the first one in the range subsection
548	float	dR_m	m	Range bin resolution in meters
552	uint32_t	nPulse	pulses/CPI	Number of pulses in one CPI
556	float	pri_s	s	Primary repetition interval in seconds

continues on next page

Table 229 – continued from previous page

Byte	C Type	Name	Unit	Description
560	uint32_t	n_associatedRPMapRangeRows	#	Number of RPMap range rows (variable packets) associated with this cookie
564	uint8_t[4]	reserved_03	N/A	Echodyne reserved field

Table 230: variable

Byte	C Type	Name	Unit	Description
0	uint32_t	rangeBinIdx	N/A	Range bin index in the range sub-section
4	uint8_t[4]	reserved_04	N/A	Echodyne reserved field
8	float[8192]	rpMapRangeRow	N/A	Range row of range-pulse map at range bin index

Cookies Packet Field Objects

beam_type value descriptions

Value	Beam Type
0	BACKGROUND_SEARCH
1	HORIZON_SEARCH
2	UNCONFIRMED_TRACK
3	CONFIRMED_TRACK
4	CLASSIFICATION
5	USER
6	DEBUG
7	ENVIRONMENT_MONITOR
8	MONOPULSE_CALIBRATION
9	RFFE_CALIBRATION
10	RFFEAF_CALIBRATION
11	USER_CALIBRATION
12	SEARCH
13	GROUND_AREA_TRACK
14	SPECTRUM_TEST

[]

5.3 10Gb Interface

This section describes the following output packet schemas:

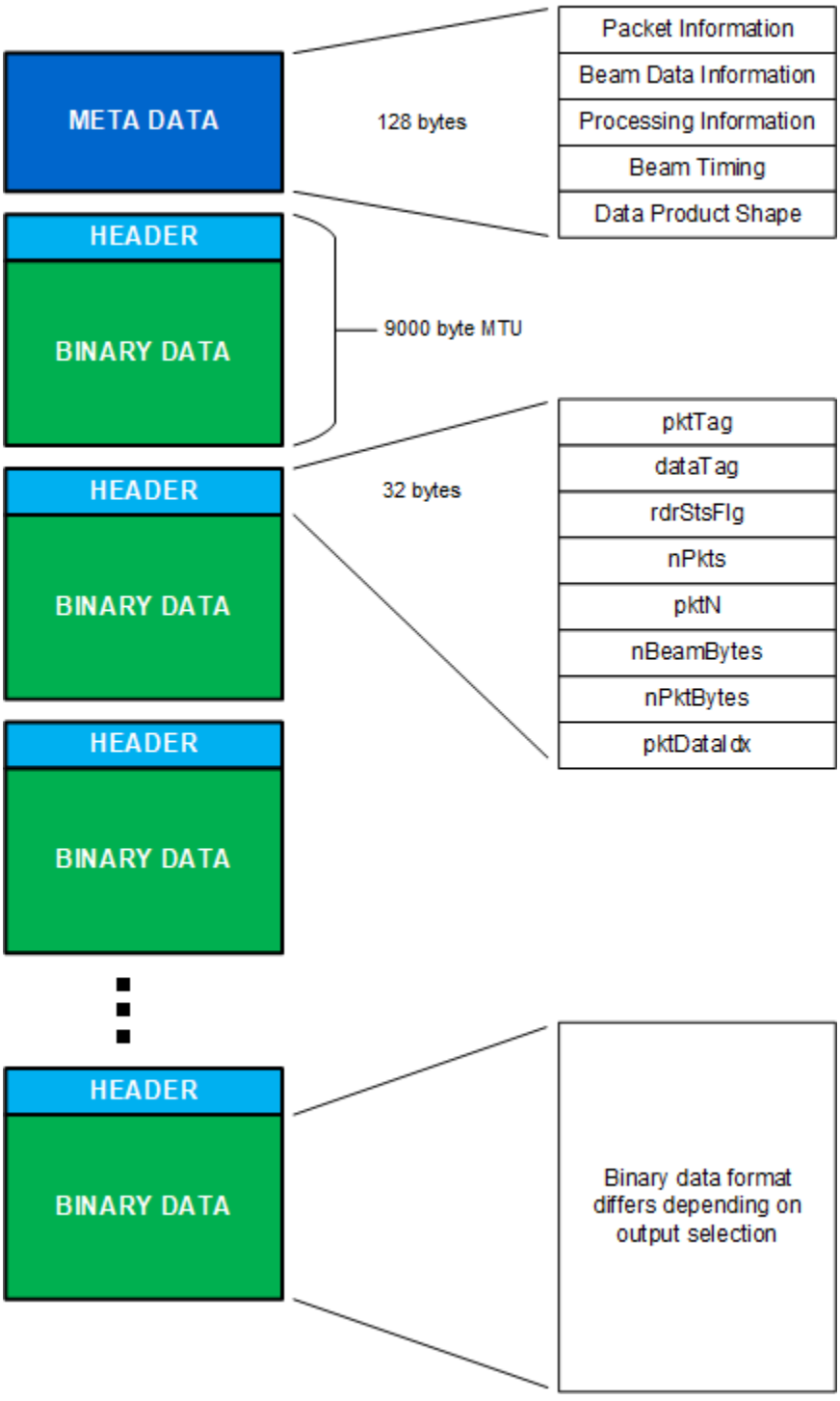
- *Rdmaps*

Rdmaps

The 10G radar data interface is used to output raw radar data in the form of Range Doppler Maps to the user. A Range Doppler Map reports the complex signal received by the radar corresponding to a given range and doppler. Packet data is in big-endian byte order.

Each reported beam will include a Metadata packet which describes the beam that has generated the information including the data product shape. Following the metadata packet area a series of data packets that comprise the RD Map data. These data packets include the total number of data packets that comprise the beam and the index of the current data packet in that sequence.

This relationship is demonstrated visually below. Packet data is in big-endian byte order.



RDMap Packet Metadata

Table 232: fixed

Byte	C Type	Name	Unit	Description
0	char[4]	packet_tag	ASCII	Packet sync word **Constraints**: ['r', 'd', 'p', 'm']
4	uint8_t[4]	reserved_00	N/A	Echodyne reserved field
8	uint64_t	beam_id	ID #	Unique ID of Beam
16	uint32_t	dsp_processing_status	bit-field	Reserved
20	uint8_t[4]	reserved_01	N/A	Echodyne reserved field
24	int64_t	exe_sec	seconds	Beam trigger time, Unix epoch seconds
32	uint32_t	exe_subsec	subseconds	Beam trigger time, this is a fractional value that can be converted to nanoseconds by y_ns = x*1e9/2^32
36	uint32_t	n_velocity_bin	velocity bins	Number of velocity bins (Doppler filters) in this beam
40	uint32_t	n_range_bin	range bins	Number of range bins in this beam
44	int32_t	n_beam_byte	bytes	Number of bytes in this beam
48	int32_t	n_beam_packet	packets	Number of UDP Datagram packets in this beam
52	uint32_t	data_descriptor	N/A	[1:0] output data format, [4:2] *output data source*
56	uint8_t[72]	reserved_block	N/A	Echodyne reserved fields

RDMap Packet Data

Table 233: fixed

Byte	C Type	Name	Unit	Description
0	char[4]	packet_tag	ASCII	Packet sync word **Constraints**: ['d', 'a', 't', 'a']
4	uint32_t	data_tag	ID #	unique header identifier of the metadata header for this beam
8	uint8_t[4]	radar_status_flag	bit-field	Reserved

continues on next page

Table 233 – continued from previous page

Byte	C Type	Name	Unit	Description
12	int32_t	n_packets	packets	The number of data packets that comprise this beam
16	int32_t	packet_num	ID #	Packet sequence number of this packet in the beam this packet belongs to
20	int32_t	n_beam_bytes	bytes	Total number of bytes in the beam this packet belongs to
24	int32_t	n_packet_bytes	bytes	The number of bytes of data in this packet
28	int32_t	packet_data_start	N/A	The start index of this packet's data in the overall *beam dataset format*

Table 234: variable

Byte	C Type	Name	Unit	Description
0	int32_t	data	N/A	Echodyne reserved fields

Glossary

output data source

0: A

1: B

2: GUARD

3: SUM

4: DIFF

output data format

0: 32-bit Complex Values (64 bits per value)

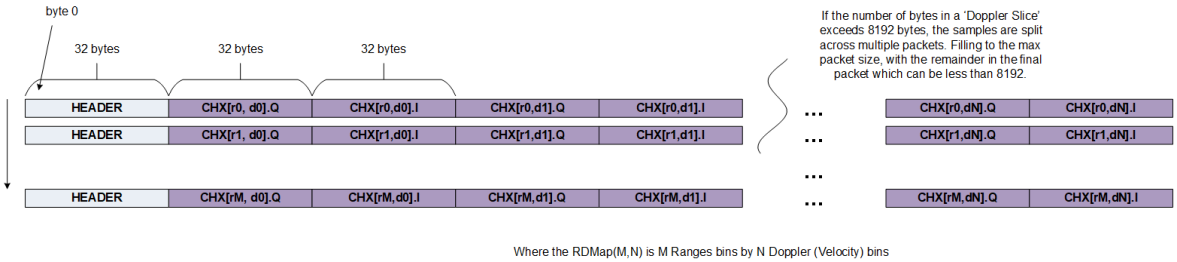
1: 32-bit Magnitude Values

2: 8-bit Log2(Magnitude) Values

beam dataset format

The following diagram illustrates the packet output stream and format of data that makes up the RDMap outputs. The RDMap data is output post doppler FFT and therefore the samples are output first along the 'Doppler Slice' dimension, then along 'Range Slice' dimension. Specifically, all samples for Range[0] for each doppler bin are output sequentially, then all samples for Range[1] for each doppler

bin, etc. Each output sample is a complex pair, with imag being output first than real. Real and imag are signed 32 bit integers output in network byte order (big-endian).



Due to MTU constraints for the 10G link, data is split into packets with a maximum number of data bytes per packet limited to 8192 bytes, plus the 32 header bytes. The design supports a maximum of 4096 doppler bins; this means that up to 4 packets may be used per doppler slice. For RDMaps with less doppler bins, only the required number of packets are used. The final packet in each doppler slice may be less than 8192 bytes.

For example, consider a RDMap of M=128 range bins and N=1050 doppler bins. This would result with the first packet being 8192 data + 32 header bytes, consisting of the first 1024 Range[0], doppler bins(0,1023). The second packet would be 208 data + 32 header bytes, with the remaining doppler bins(1024-1049) for Range[0]. This would be followed by an 8192 data + 32 header bytes which is the start of the next range bin[1], doppler bins(0,1023), and so on. This results in a total of 1 metatdata packet followed by 256 data packets (2 packets per range bin).

6 Initiated Built-in Test (IBIT)

Initiated Built-in Tests are used to diagnose and troubleshoot radar operation.

6.1 Antenna Diode Check

The *ibit_antenna_diode_check()* request instructs the radar to perform a test on the antenna diodes. The results are returned in a nested list structure that represents an array that is 17 by 8 by 16 integers. Each integer is a bitmap represented in decimal. Note that the row and column are 1 based in their numbering even though the array is generally considered 0 based in JSON. The driver, however, is 0 based.

Interpreting the Output of *ibit_antenna_diode_check* Command

The annotated "*faults*" output below shows how to interpret the arrays that come out of the *ibit_antenna_diode_check* response.:

```
"faults": [
    [ # Column 1 (note that columns are 1 based)
      #  (Note that rows are 1 based)
      #  +------ Driver 0 (note that drivers are 0 based)
      #  |
      #  |  + -- Driver 1         ...
      #  |  |                                        +-- Driver 15
      #  v  v                                        v
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Row 1
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Row 2
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Row 3
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Row 4
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Row 5
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Row 6
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Row 7
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  # Row 8
    ],
    [ # Column 2
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Row 1
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Row 2
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Row 3
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Row 4
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Row 5
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Row 6
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Row 7
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  # Row 8
    ],
```

(continues on next page)

(continued from previous page)

```
    [ # Column 3
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    ],
    [ # Column 4
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    ],
    [ # Column 5
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    ],
    [ # Column 6
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    ],
    [ # Column 7
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
```

(continues on next page)

(continued from previous page)

```
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    ],
    [ # Column 8
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    ],
    [ # Column 9
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    ],
    [ # Column 10
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    ],
    [ # Column 11
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 2, 0, 0, 2, 0, 0, 2, 0, 0, 2, 0, 0],
```

(continues on next page)

(continued from previous page)

```
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    ],
    [ # Column 12
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    ],
    [ # Column 13
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    ],
    [ # Column 14
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    ],
    [ # Column 15
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

(continues on next page)

(continued from previous page)

```
    ],
    [ # Column 16
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    ],
    [ # Column 17
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Row 1
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Row 2
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Row 3
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Row 4
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Row 5
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Row 6
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], # Row 7
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  # Row 8
    ]
]
```

Intepreting the Driver Integers

The integers in the inner most list represent a bitmap of the possible faults. This table indicates which faults are represented by the bits 1, 2, 4, and 8.

bit	Meaning
1	Short to Vneg
2	Short to Vss
4	Short to Vcc or Vhv
8	Short to drive

The most common value is 0 which means that the diode has no faults.

The example *faults* output shows some faults.

Columns 10 and 11 have multiple faults. The 1 indicates a short to VNeg was detected. The 2 indicates a short to Vss was detected.

Column 13, row 3, driver 13 also has a single fault.

See them here:

```
[ # Column 10
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
],
[ # Column 11
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 2, 0, 0, 2, 0, 0, 2, 0, 0, 2, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
],
...
[ # Column 13
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
],
```

7 Startup Built-in Test (SBIT)

Startup Built-in Tests are used to diagnose and troubleshoot radar operation and are run when radar firmware is started.

sbit_bitmap index	Fault Identifier
[0][0]	MAPCOM_EEPROM_FAULT
[0][1]	ELASPED_TIMER_FAULT
[0][2]	MAPCOM_6524_A_FAULT
[0][3]	MAPCOM_6524_B_FAULT
[0][4]	MAPCOM_90160_FAULT
[0][5]	IMU_FAULT
[0][6]	MAPCOM_468_FAULT
[0][7]	ANTENNA_468_FAULT
[0][8]	SYNTH_4372_FAULT
[0][9]	CLKDIV_7044_FAULT
[0][10]	CLK_GEN_5332_FAULT
[0][11]	MXFE_FAULT
[0][12]	GNSS_FAULT
[0][13]	UDCX_EEPROM_FAULT
[0][14]	UDCX_SYNTH_6948_A_FAULT
[0][15]	SBIT_RESERVED_0
[0][16]	UDCX_90160_FAULT
[0][17]	UDCX_QUAD_POT_4461_A_FAULT
[0][18]	UDCX_QUAD_POT_4461_B_FAULT
[0][19]	UDCX_QUAD_POT_4461_1A_FAULT
[0][20]	UDCX_QUAD_POT_4461_1B_FAULT
[0][21]	UDCX_PORT_EXP_6524_FAULT
[0][22]	UDCX_TEMP_468_FAULT
[0][23]	UDCX_VDOM_FAULT
[0][24]	TX_EEPROM_FAULT
[0][25]	TX_90160_FAULT
[0][26]	TX_DUAL_POT_4661_FAULT
[0][27]	TX_QUAD_POT_4461_FAULT
[0][28]	TX_6416A_FAULT
[0][29]	CLK_GEN_5341_FAULT
[0][30]	FPGA_10GbE_FAULT
[0][31]	FPGA_40GbE_FAULT
[0][32]	ANTENNA_FLASH_FAULT
[0][33]	UDCX_PORT_EXP_6416_FAULT
[0][34]	UDCX_ADC_7291_A_FAULT
[0][35]	UDCX_ADC_7291_B_FAULT
[0][36]	TX_468_FAULT

continues on next page

Table 235 – continued from previous page

sbit_bitmap index	Fault Identifier
[0][37]	TX_ADC0_FAULT
[0][38]	TX_ADC1_FAULT
[0][39]	FPGA_DRAM
[0][40]	ANTENNA_FLASH_INT
[0][41]	ANTENNA_ASIC
[0][42]	ANTENNA_CBIT_INT
[0][43]	OCXO_DAC_FAULT

8 Continuous Built-in Test (CBIT)

Continuous Built-in Tests are used to diagnose and troubleshoot radar operation and are run at the CBIT interval, 2 seconds. Each fault or warning with _LOCAL or a number appended corresponds to a particular temperature, voltage or current. LOCAL corresponds to the first temperature, voltage, etc in each array and the numbers correspond to the following values. LOCAL corresponds to the temperature at the chip itself, the rest are sensors in other locations.

cbit_bitmap index	Fault/Warning Identifier
[0][0]	MAPCOM_TEMP_WARNING_LOCAL
[0][1]	MAPCOM_TEMP_WARNING_1
[0][2]	MAPCOM_TEMP_WARNING_2
[0][3]	MAPCOM_TEMP_WARNING_3
[0][4]	MAPCOM_TEMP_WARNING_4
[0][5]	MAPCOM_TEMP_WARNING_5
[0][6]	MAPCOM_TEMP_WARNING_6
[0][7]	MAPCOM_TEMP_WARNING_7
[0][8]	MAPCOM_TEMP_WARNING_8
[0][9]	MAPCOM_TEMP_FAULT_LOCAL
[0][10]	MAPCOM_TEMP_FAULT_1
[0][11]	MAPCOM_TEMP_FAULT_2
[0][12]	MAPCOM_TEMP_FAULT_3
[0][13]	MAPCOM_TEMP_FAULT_4
[0][14]	MAPCOM_TEMP_FAULT_5
[0][15]	MAPCOM_TEMP_FAULT_6
[0][16]	MAPCOM_TEMP_FAULT_7
[0][17]	MAPCOM_TEMP_FAULT_8
[0][18]	ANTENNA_TEMP_WARNING_LOCAL
[0][19]	ANTENNA_TEMP_WARNING_1
[0][20]	ANTENNA_TEMP_WARNING_2
[0][21]	ANTENNA_TEMP_WARNING_3

continues on next page

Table 236 – continued from previous page

cbit_bitmap index	Fault/Warning Identifier
[0][22]	ANTENNA_TEMP_WARNING_4
[0][23]	ANTENNA_TEMP_WARNING_5
[0][24]	ANTENNA_TEMP_WARNING_6
[0][25]	ANTENNA_TEMP_WARNING_7
[0][26]	ANTENNA_TEMP_WARNING_8
[0][27]	ANTENNA_TEMP_FAULT_LOCAL
[0][28]	ANTENNA_TEMP_FAULT_1
[0][29]	ANTENNA_TEMP_FAULT_2
[0][30]	ANTENNA_TEMP_FAULT_3
[0][31]	ANTENNA_TEMP_FAULT_4
[0][32]	ANTENNA_TEMP_FAULT_5
[0][33]	ANTENNA_TEMP_FAULT_6
[0][34]	ANTENNA_TEMP_FAULT_7
[0][35]	ANTENNA_TEMP_FAULT_8
[0][36]	UDCX_TEMP_WARNING_LOCAL
[0][37]	UDCX_TEMP_WARNING_1
[0][38]	UDCX_TEMP_WARNING_2
[0][39]	UDCX_TEMP_WARNING_3
[0][40]	UDCX_TEMP_WARNING_4
[0][41]	UDCX_TEMP_WARNING_5
[0][42]	UDCX_TEMP_WARNING_6
[0][43]	UDCX_TEMP_WARNING_7
[0][44]	UDCX_TEMP_WARNING_8
[0][45]	UDCX_TEMP_FAULT_LOCAL
[0][46]	UDCX_TEMP_FAULT_1
[0][47]	UDCX_TEMP_FAULT_2
[0][48]	UDCX_TEMP_FAULT_3
[0][49]	UDCX_TEMP_FAULT_4
[0][50]	UDCX_TEMP_FAULT_5
[0][51]	UDCX_TEMP_FAULT_6
[0][52]	UDCX_TEMP_FAULT_7
[0][53]	UDCX_TEMP_FAULT_8
[0][54]	SYNTH_4372_FAULT
[0][55]	CLK_DIV_7044_FAULT
[0][56]	CLK_GEN_5332_FAULT
[0][57]	MXFE_FAULT
[0][58]	SYNTH_6948_A_FAULT
[0][59]	CBIT_RESERVED_0
[0][60]	VDOM_UNIT_FAULT
[0][61]	PART_BOOT_WARNING
[0][62]	PART_ROOT_A_WARNING
[0][63]	PART_ROOT_B_WARNING

continues on next page

Table 236 – continued from previous page

cbit_bitmap index	Fault/Warning Identifier
[1][0]	PART_FW_A_WARNING
[1][1]	PART_FW_B_WARNING
[1][2]	PART_LOG_WARNING
[1][3]	PART_USER_WARNING
[1][4]	PART_MFG_WARNING
[1][5]	PART_JAXA_WARNING
[1][6]	PART_BOOT_FAULT
[1][7]	PART_ROOT_A_FAULT
[1][8]	PART_ROOT_B_FAULT
[1][9]	PART_FW_A_FAULT
[1][10]	PART_FW_B_FAULT
[1][11]	PART_LOG_FAULT
[1][12]	PART_USER_FAULT
[1][13]	PART_MFG_FAULT
[1][14]	PART_JAXA_FAULT
[1][15]	MEMORY_WARNING
[1][16]	MEMORY_FAULT
[1][17]	TX0_TEMP_WARNING_LOCAL
[1][18]	TX0_TEMP_WARNING_1
[1][19]	TX0_TEMP_WARNING_2
[1][20]	TX0_TEMP_WARNING_3
[1][21]	TX0_TEMP_WARNING_4
[1][22]	RESERVED
[1][23]	RESERVED
[1][24]	RESERVED
[1][25]	RESERVED
[1][26]	TX0_TEMP_FAULT_LOCAL
[1][27]	TX0_TEMP_FAULT_1
[1][28]	TX0_TEMP_FAULT_2
[1][29]	TX0_TEMP_FAULT_3
[1][30]	TX0_TEMP_FAULT_4
[1][31]	RESERVED
[1][32]	RESERVED
[1][33]	RESERVED
[1][34]	RESERVED
[1][35]	TX1_TEMP_WARNING_LOCAL
[1][36]	TX1_TEMP_WARNING_1
[1][37]	TX1_TEMP_WARNING_2
[1][38]	TX1_TEMP_WARNING_3
[1][39]	TX1_TEMP_WARNING_4
[1][40]	RESERVED
[1][41]	RESERVED

continues on next page

Table 236 – continued from previous page

cbit_bitmap index	Fault/Warning Identifier
[1][42]	RESERVED
[1][43]	RESERVED
[1][44]	TX1_TEMP_FAULT_LOCAL
[1][45]	TX1_TEMP_FAULT_1
[1][46]	TX1_TEMP_FAULT_2
[1][47]	TX1_TEMP_FAULT_3
[1][48]	TX1_TEMP_FAULT_4
[1][49]	RESERVED
[1][50]	RESERVED
[1][51]	RESERVED
[1][52]	PPS_EXT_REF_WARNING
[1][53]	TX0_ADC0_OVER_VOLTAGE_FAULT_0
[1][54]	TX0_ADC0_OVER_VOLTAGE_FAULT_1
[1][55]	TX0_ADC0_OVER_VOLTAGE_FAULT_2
[1][56]	TX0_ADC0_OVER_VOLTAGE_FAULT_3
[1][57]	TX0_ADC0_UNDER_VOLTAGE_FAULT_0
[1][58]	TX0_ADC0_UNDER_VOLTAGE_FAULT_1
[1][59]	TX0_ADC0_UNDER_VOLTAGE_FAULT_2
[1][60]	TX0_ADC0_UNDER_VOLTAGE_FAULT_3
[1][61]	TX0_ADC0_OVER_CURRENT_FAULT_0
[1][62]	TX0_ADC0_OVER_CURRENT_FAULT_1
[1][63]	TX0_ADC0_OVER_CURRENT_FAULT_2
[2][0]	TX0_ADC0_OVER_CURRENT_FAULT_3
[2][1]	TX0_ADC1_OVER_CURRENT_FAULT_0
[2][2]	TX0_ADC1_OVER_CURRENT_FAULT_1
[2][3]	TX0_ADC1_OVER_VOLTAGE_FAULT_0
[2][4]	TX0_ADC1_OVER_VOLTAGE_FAULT_1
[2][5]	TX0_ADC1_OVER_VOLTAGE_FAULT_2
[2][6]	TX0_ADC1_OVER_VOLTAGE_FAULT_3
[2][7]	TX0_ADC1_UNDER_VOLTAGE_FAULT_0
[2][8]	TX0_ADC1_UNDER_VOLTAGE_FAULT_1
[2][9]	TX0_ADC1_UNDER_VOLTAGE_FAULT_2
[2][10]	TX0_ADC1_UNDER_VOLTAGE_FAULT_3
[2][11]	TX0_ADC1_OVER_TEMPERATURE_FAULT_0
[2][12]	TX0_ADC1_OVER_TEMPERATURE_FAULT_1
[2][13]	TX1_ADC0_OVER_VOLTAGE_FAULT_0
[2][14]	TX1_ADC0_OVER_VOLTAGE_FAULT_1
[2][15]	TX1_ADC0_OVER_VOLTAGE_FAULT_2
[2][16]	TX1_ADC0_OVER_VOLTAGE_FAULT_3
[2][17]	TX1_ADC0_UNDER_VOLTAGE_FAULT_0
[2][18]	TX1_ADC0_UNDER_VOLTAGE_FAULT_1
[2][19]	TX1_ADC0_UNDER_VOLTAGE_FAULT_2

continues on next page

Table 236 – continued from previous page

cbit_bitmap index	Fault/Warning Identifier
[2][20]	TX1_ADC0_UNDER_VOLTAGE_FAULT_3
[2][21]	TX1_ADC0_OVER_CURRENT_FAULT_0
[2][22]	TX1_ADC0_OVER_CURRENT_FAULT_1
[2][23]	TX1_ADC0_OVER_CURRENT_FAULT_2
[2][24]	TX1_ADC0_OVER_CURRENT_FAULT_3
[2][25]	TX1_ADC1_OVER_CURRENT_FAULT_0
[2][26]	TX1_ADC1_OVER_CURRENT_FAULT_1
[2][27]	TX1_ADC1_OVER_VOLTAGE_FAULT_0
[2][28]	TX1_ADC1_OVER_VOLTAGE_FAULT_1
[2][29]	TX1_ADC1_OVER_VOLTAGE_FAULT_2
[2][30]	TX1_ADC1_OVER_VOLTAGE_FAULT_3
[2][31]	TX1_ADC1_UNDER_VOLTAGE_FAULT_0
[2][32]	TX1_ADC1_UNDER_VOLTAGE_FAULT_1
[2][33]	TX1_ADC1_UNDER_VOLTAGE_FAULT_2
[2][34]	TX1_ADC1_UNDER_VOLTAGE_FAULT_3
[2][35]	TX1_ADC1_OVER_TEMPERATURE_FAULT_0
[2][36]	TX1_ADC1_OVER_TEMPERATURE_FAULT_1
[2][37]	UDCX_ADC0_UNDER_VOLTAGE_FAULT_0
[2][38]	UDCX_ADC0_UNDER_VOLTAGE_FAULT_1
[2][39]	UDCX_ADC0_UNDER_VOLTAGE_FAULT_2
[2][40]	UDCX_ADC0_UNDER_VOLTAGE_FAULT_3
[2][41]	UDCX_ADC0_UNDER_VOLTAGE_FAULT_4
[2][42]	UDCX_ADC0_UNDER_VOLTAGE_FAULT_5
[2][43]	UDCX_ADC0_UNDER_VOLTAGE_FAULT_6
[2][44]	UDCX_ADC0_OVER_VOLTAGE_FAULT_0
[2][45]	UDCX_ADC0_OVER_VOLTAGE_FAULT_1
[2][46]	UDCX_ADC0_OVER_VOLTAGE_FAULT_2
[2][47]	UDCX_ADC0_OVER_VOLTAGE_FAULT_3
[2][48]	UDCX_ADC0_OVER_VOLTAGE_FAULT_4
[2][49]	UDCX_ADC0_OVER_VOLTAGE_FAULT_5
[2][50]	UDCX_ADC0_OVER_VOLTAGE_FAULT_6
[2][51]	UDCX_ADC1_OVER_CURRENT_FAULT_0
[2][52]	UDCX_ADC1_OVER_CURRENT_FAULT_1
[2][53]	UDCX_ADC1_OVER_CURRENT_FAULT_2
[2][54]	UDCX_ADC1_OVER_CURRENT_FAULT_3
[2][55]	UDCX_ADC1_OVER_CURRENT_FAULT_4
[2][56]	UDCX_ADC1_OVER_CURRENT_FAULT_5
[2][57]	UDCX_ADC1_UNDER_VOLTAGE_FAULT_6
[2][58]	UDCX_ADC1_UNDER_VOLTAGE_FAULT_7
[2][59]	UDCX_ADC1_OVER_VOLTAGE_FAULT_6
[2][60]	UDCX_ADC1_OVER_VOLTAGE_FAULT_7

continues on next page

Table 236 – continued from previous page

cbit_bitmap index	Fault/Warning Identifier
[2][61]	FPGA_10G_LINK_SPEED_FAULT

9 Kinematics Setup Guide

9.1 Introduction

Because the radar uses the placement of terrain in its field of view to optimize its search and track behavior, it will not operate correctly unless it is made aware of its global position and orientation. A user is therefore required to give it that information using one of the set_x_kinematics Radar App Commands before the radar can be operated.

This documentation uses the term **kinematics** to refer to the position, orientation, linear velocity, and angular velocity of the radar; however, this chapter will only apply to cases where linear and angular velocity are zero. This chapter will also deal with setting up a single radar using Echoshield's API. For all other cases, and for further explanations, please refer to later chapters.

9.2 Setting Kinematics Before Radaring

The radar won't operate if it is not told its kinematics, so a call to one of the kinematics setters is required before *mode_set_start* can be called. The best way to do this manually is the *set_radar_kinematics* function, but *set_reference_kinematics_ecef* and *set_reference_kinematics_geodetic* are also available. This chapter of the doc will only cover the use of *set_radar_kinematics*; to learn about the other setters please refer to *Echoshield Coordinate Frame Definitions* and *Reference Frame Guide*.

In general, the kinematics can be set in any radar mode; therefore, the initial kinematics setup can be done in either service or idle states.

9.3 Using the set_radar_kinematics Command

To inform the radar of its location and orientation in the world, the *set_radar_kinematics* function takes seven arguments:

- **latitude_deg** in degrees
- **longitude_deg** in degrees
- **altitude_m** in meters above mean sea level or above the WGS84 ellipsoid (see the **altitude-type** argument below).
- **heading_deg** in degrees from north. It is measured clockwise so that a heading of 90 degrees will point due east.

- **pitchback_deg** in degrees. This is the angle between the broadside beam and the level plane at the radar's location. It is oriented so that a positive angle points the radar above the horizon.
- **tilt_deg** in degrees. This is a left-handed rotation around the radar's 'broadside' or z-axis. In other words, if you were standing and facing with your chest oriented in the same direction as the radar, a positive tilt would move your left shoulder down and right shoulder up. A user should always try to level the radar and minimize tilt.
- **altitude-type** modifies the **altitude_m** to mean either the number of meters above mean sea level as defined by the EGM96 model (also referred to as a geoid) with the parameter "orthometric", or the number of meters above the WGS84 ellipsoid with the parameter "ellipsoidal". The difference between these two measurements can be as great as 90 meters, so it is important that it matches the sensor.

It should be noted that the heading, pitchback, and tilt are calculated as Euler angles, meaning they should be measured in the order heading, pitchback, tilt. A sensor may output orientation that is derived in a slightly different way, so an engineer debugging the system may want to confirm that the angles they are inputting are defined the same way.

Also, this command assumes that the radar is stationary. For nonzero velocity, please refer to *set_reference_kinematics_ecef*.

9.4 Checking the Current Kinematics

There are several ways to check the radar's kinematics after they've been set. Two such commands are recommended after using *set_radar_kinematics*:

- *get_radar_kinematics*: this returns the kinematics the radar is currently using, and should return exactly what was set with *set_radar_kinematics*.
- *get_ins_data*: outputs what the internal sensors measure for kinematics. If the INS solution is valid, it should report values that are close to those reported by *get_radar_kinematics*.

The kinematics commands also report a "kinematics_quality" response, which declares how well the user-set kinematics align with the onboard INS's kinematics solution. This information is also available in the 1G Status output packet: *Status*.

The INS is subject to a few limitations. It should not be used when the radar is lying on its back or front, as pitchback measurements close to 90 degrees will result in numerical instability from gimbal lock. Also, the INS relies on a good GPS connection and will not report a valid solution if the GPS is blocked.

9.5 Streaming Kinematics

In the case where kinematics needs to be repeatedly updated over time (such as when the radar platform moves), a *kinematics_input_port* UDP port is made available that accepts kinematics setter commands. This allows a user to repeatedly update kinematics without tying up the comand port.

The commands sent to the *kinematics_input_port* port should be in the same JSON format as any command sent to the radar command port. Accepted commands are:

- *set_radar_kinematics*
- *set_reference_kinematics_ecef*
- *set_reference_kinematics_geodetic*

While any of the three commands are accepted through the kinematics port, note that *set_radar_kinematics* and *set_reference_kinematics_geodetic* only support use cases where the radar frame has zero-velocity with respect to the ECEF frame. In any case where the radar frame has a non zero-velocity with respect to the world, the *set_reference_kinematics_ecef* command should be used.

The radar will accept kinematics input at rates up to 125 commands per second. If commands are sent at a higher rate, radar kinematics will be set at 125Hz using only the most recent command received per update period.

The `id` field of the these commands should be incremented for each successive message. If the radar receives a command with an `id` value that is not an increment of the last one received, it will issue a warning to its syslog.

For each command sent to the *kinematics_input_port*, a kinematics packet will be generated and output on the *kinematics_port*. Users are encouraged to monitor the kinematics output port to ensure that radar kinematics are being updated as they expect.

9.6 Measuring and Setting Kinematics with the Sunsight AAT

Echodyne uses a Sunsight™ Antenna Alignment Tool (AAT) to measure the location and orientation of EchoShield radars used in stationary applications.

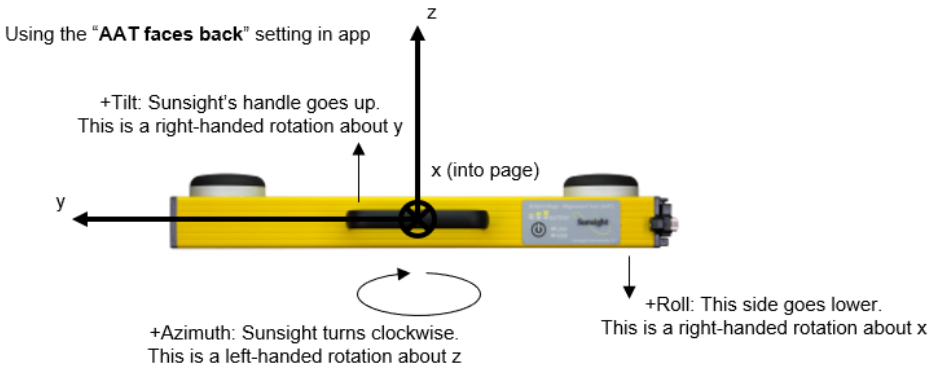


Fig. 1: Sunsight AAT azimuth, tilt, and roll directions and axes with the **AAT faces back** setting in app.

The AAT reports location in standard geodetic latitude, longitude, and height-above-mean-sea-level (A.K.A. **orthometric**) altitude coordinates, but it reports orientation in a custom format. The AAT reports three Euler angles for four different AAT mountings. Use the **AAT faces back** setting. Using a different setting will rotate the nominal AAT reference frame, and cause the conversions to heading, pitchback, and tilt to change. For this setting, the reported angles are:

Azimuth - The left-handed rotation about the AAT's z-axis (with zero azimuth when the AAT's x-axis points South). This is 180 degrees from the radar's heading when mounted using the bracket.

Tilt - The right-handed rotation about the AAT's y-axis with zero pitch when the AAT's x-axis is level (in the East-North plane).

Roll - The right-handed rotation about the AAT's x-axis with zero roll when the AAT's y-axis is level (in the East-North plane).

These measurements correspond with the heading_deg, pitchback_deg, and tilt_deg definitions, respectively, used by the *set_radar_kinematics* command (note that the command's tilt_deg correspond with the AAT's Roll measurement, and not the AAT's Tilt measurement). The AAT should measure 180 degrees Azimuth when the AAT's x-axis points south and the radar points north, if the AAT is attached to the radar using the mounting bracket. If the AAT is measuring using its **AAT faces back** configuration a user can submit these orientation angles into the *set_radar_kinematics*, with only the heading requiring alteration.

In summary,

1. Use *Sunsight faces back* setting in the Sunsight app.
2. Issue the *set_radar_kinematics* command with the following parameters:
    - latitude_deg = $\text{latitude}_{\text{AAT}}$
    - longitude_deg = $\text{longitude}_{\text{AAT}}$
    - altitude_m = $\text{altitude}_{\text{AAT}}$
    - heading_deg = $\text{azimuth}_{\text{AAT}}$ - 180
    - pitchback_deg = tilt_{AAT}

- tilt_deg = roll_{AAT}
- altitude_type = orthometric

10 Echoshield Coordinate Frame Definitions

10.1 Introduction

An EchoShield radar can represent the state of its **kinematics** (position, orientation, linear velocity, and rotational velocity) in several coordinate frames. This section defines those frames, along with some intended uses for each.

Echoshield derives its coordinate frame definitions using earth models defined by the WGS84 ellipsoid and the EGM96 geoid models.

10.2 Radar Frame

The radar frame refers to a coordinate frame that is centered on the face of the radar. The x-axis is defined to come out the side of the radar, the y-axis out the top, and the z-axis out the front face, or 'broadside', to complete the right-handed triad. 'Right' and 'left' refer to negative and positive x, respectively.

When talking about radar-frame Euler angles, the yaw, pitch, and roll angles are applied in that order and are about the z, y, and x axes respectively.

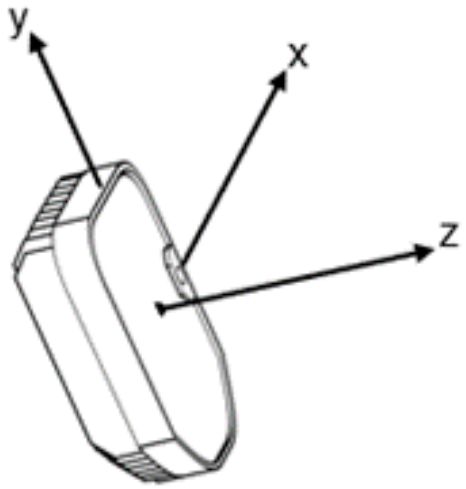


Fig. 2: The radar frame with its Cartesian triad.

The radar's antenna frame (also known as Azimuth-Elevation frame) has the same origin on the face of the radar, but Azimuth and Elevation are degrees measured from broadside with positive defined as up (or towards positive y) and left (towards positive x), respectively.

Similarly, UV coordinates generally are defined using radar face as origin and broadside as zero-u and zero-v.

10.3 ECEF Frame

The ECEF (Earth-Centered, Earth-Fixed) frame is the most significant non-radar frame, as it is treated as the basis upon which all other frames are defined.

It is a 3D Cartesian coordinate frame that is fixed to the rotating Earth. It is defined with the origin at the center of the earth, x-axis extending through the prime meridian and equator (or 0 degrees latitude, 0 degrees longitude), the y-axis extending through 90 degrees East and the equator, and the z-axis through the north pole.

EchoShield's ECEF model is as defined by the WGS84 earth model, meaning a given ECEF coordinate will have a unique latitude, longitude and height-above-geoid. The radar's kinematics treat the ECEF as a non-inertial or fixed frame.

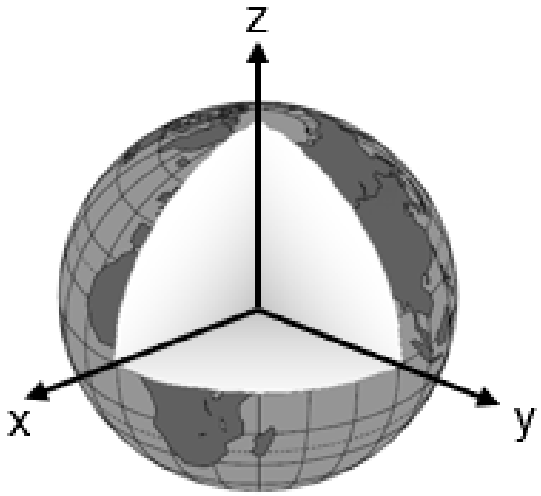


Fig. 3: The Earth-centered, Earth-fixed frame.

10.4 ENU Frame

The ENU (east-north-up) is a locally-defined 3D frame with its origin defined with a latitude and longitude. From its origin, the x, y, and z axes point due east, due north, and up, respectively. Up is defined as the normal direction to the WGS84 ellipsoid.

While the ENU frame can be centered in an arbitrary spot, ENU velocities tend to be relative to the world, unless otherwise noted. This means that a moving radar's ENU frame will be centered on the radar, but the velocities will be with respect to the stationary ground.

Because the directions of east, north, and up change depending on where in the world the origin is, a user has to be careful not to use coordinates from two different ENU frames without careful conversion. For instance,

if a target's coordinates are found in an ENU frame that is centered on a radar, then it would be incorrect to feed that information to a component that is expecting ENU centered on the user.

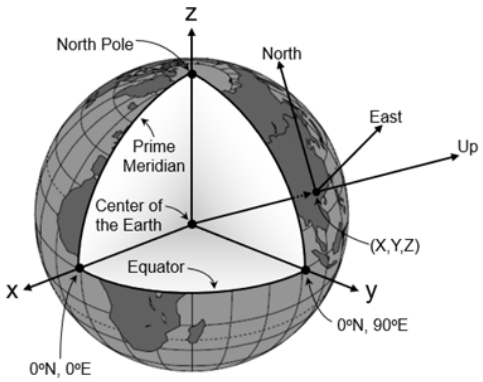


Fig. 4: Lines of Latitude and Longitude are drawn on a globe. cut-away shows the X, Y, and Z axes of the Earth Centered Earth Fixed coordinate frame. An East-North-Up coordinate frame is shown for a particular, arbitrary, (X, Y, Z) location.

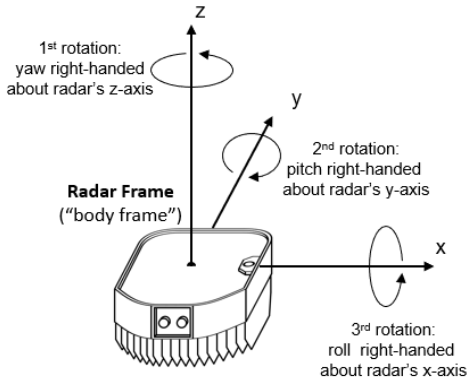
10.5 Geodetic Frame

The term "geodetic" refers to the set of latitude, longitude, altitude, and right-handed Euler angles with respect to the local ENU frame. While the latitude and longitude have concrete definitions (per WGS84), altitude can be defined with respect to the ellipsoid or the geoid (see below), and Euler angles can have many definitions.

10.6 Euler Angle (Yaw, Pitch and Roll) Conventions

Yaw, pitch, and roll are defined as being rotations around a body's z, y, and x axes respectively. Euler angles are found by first aligning two coordinate frames, then making the three rotations. EchoShield uses a YPR convention of **right-handed** rotations applied **intrinsically** (meaning the rotations are applied to the rotated body's coordinate axes) in a **Z-Y-X order** (meaning the rotation is achieved by yawing around the z-axis, pitching around y, and then rolling around x).

One consequence of this is that when finding Euler angles with respect to the ENU frame, a zero yaw, pitch and roll means the radar is lying flat on its back. A standard orientation frequently seen in the field, with the radar upright and broadside 20 degrees above the horizon, will give it 70 degrees of roll.



10.7 Heading, Pitchback and Tilt

The words yaw, pitch, and roll are overloaded, and Euler angles can be difficult to calculate. One way of limiting confusion is by defining heading, pitchback, and tilt, to align with a user's intuition of orientation. These angles are the best way for a user to quickly set up the radar, using the *set_radar_kinematics* command.

Heading is defined with zero pointing due north. It is measured clockwise so that a heading of 90 degrees will point the radar due east.

Pitchback is the angle between the radar's broadside or z-axis beam and its local east-north plane. It is oriented so that a positive angle points the radar above the horizon.

Tilt rotates around the radar's broadside axis. It is a left-handed rotation around broadside, so if you were standing and facing with your chest oriented in the same direction as the radar, a positive tilt would move your left shoulder down and right shoulder up.

These angles are standard Euler angles as defined above, but converted to match these definitions. If an engineer were to calculate these angles from a sensor, they would first find yaw, pitch, and roll with respect to the ENU frame as a starting point.

- Heading would equal the inverse (negative) of the yaw with respect to ENU, plus 180 degrees.
- Tilt would equal the inverse of the pitch with respect to ENU.
- Pitchback would equal the inverse of the roll with respect to ENU, plus 90 degrees.

However, it should be noted that these angles are for manual entry, and if detailed sensor data is available it is recommended to set the kinematics using the reference frame geodetic or ECEF radar commands.

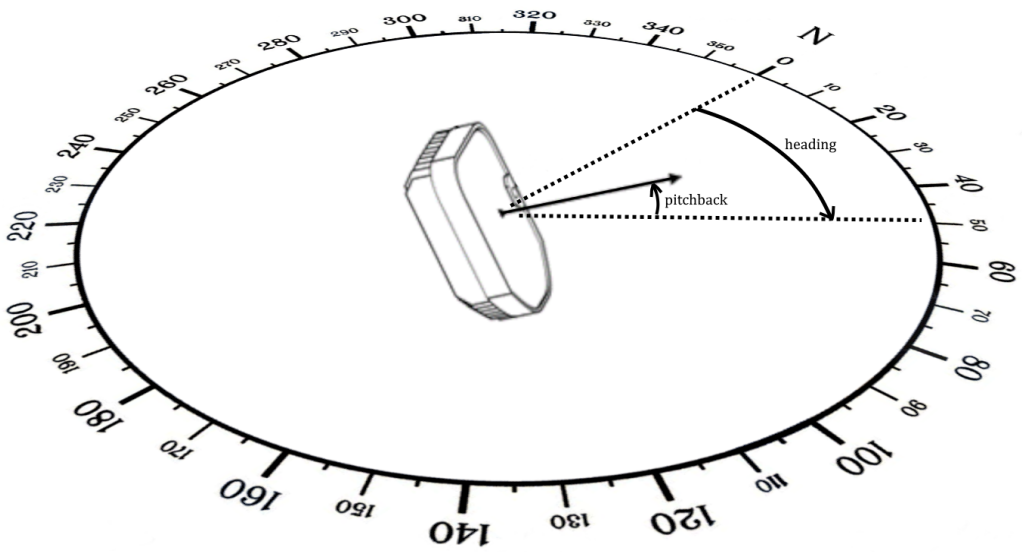


Fig. 5: An example of a radar with 50 degrees of heading, about 20 degrees pitchback, and zero tilt.

10.8 Reference Frame

The reference frame is a tool for users of multi-radar or moving radar installations to more quickly apply sensor data to the radar. A user can define offsets between the reference frame and the radar frame, and with the assumption that these offsets are unchanging, apply sensor data to the reference frame without needing to convert to the radar frame.

For more information, please refer to the *Reference Frame Guide*.

10.9 Quaternions

Quaternions are a way of representing rotations, an alternative to Euler angles. They are imaginary numbers with one real and three imaginary components, usually labelled w, x, y and z, respectively.

While they are mathematically powerful, they aren't useful to most people as a visualization tool. For this reason, most commands that offer quaternions as an input or output will have a complimentary command that offers Euler angles (for example, *get_radar_kinematics_ecef*, which returns a quaternion for rotations, and *get_radar_kinematics_geodetic*, which returns Euler angles w.r.t. ENU).

10.10 Altitude Relative Ellipsoid and Geoid

There are two common ways to measure altitude of an object: the object's height relative to the *geoid* and its height relative to the *ellipsoid*. In EchoShield, these are defined as:

Ellipsoid refers to an ellipsoid as defined by the WGS84 earth model. It is a close approximation of the shape of a deformed earth, but it is a simple surface and doesn't reflect topography, subsurface densities, or other qualiites that can have wide ranging impacts on targets, measurement devices, and human perception.

Geoid refers to the shape created by the EGM96 data model that attempts to quantify the sea level at each point on the globe (many sources refer to it as Mean Sea Level), deriving the sea level from gravimetric data. This is not a regular shape. It is a very lumpy sphere.

Most applications will default to presenting and using altitudes that are relative to the geoid, but will also allow a user to specify which to use. This is important to pay attention to, as the difference between the two approximations can be as much as 90 meters.

11 Reference Frame Guide

11.1 Overview

EchoShield includes a suite of tools enabling multi-radar and nonstationary radar systems. One of these tools is the Reference Frame, which is one method of setting the position and orientation (the **kinematics**) of the radar.

The reference frame is an extra Cartesian coordinate frame with which a user can input kinematics. It is defined using static offsets, meaning that the radar frame and the reference frame are stationary with respect to one another. Once a user has correctly calculated the position and rotation of the radar relative to the reference frame, they can set the kinematics of the reference frame and the radar frame will automatically update to its correct position.

This is useful because position and orientation sensors, such as inertial INS units, are rarely colocated, or even defined with the same coordinate conventions, as a radar. And if a second radar is used, then a user can simply find a second set of offsets, and then use one sensor measurement to update the kinematics of both radars.

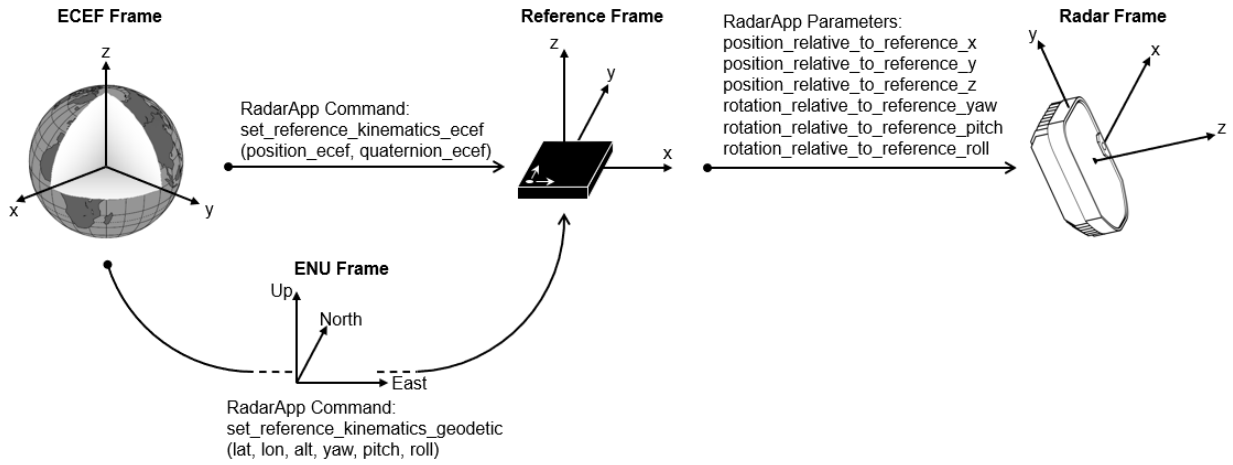


Fig. 6: The different coordinate frames involved in specifying the radar's global position, orientation, velocity, and rotation rate. When using the set_reference_kinematics_geodetic() command there is an implicit use of a local ENU Frame at the specified latitude and longitude to which the yaw, pitch, and roll rotations are referenced.

This chapter will outline the usage of the reference frame and give instructions for how it can be used correctly.

11.2 Setting the Reference kinematics

There are four commands a user can use to set and retrieve the reference kinematics:

- *set_reference_kinematics_ecef*, which takes position, quaternion orientation, linear velocity, and rotational velocity, all with respect to ECEF.
- *set_reference_kinematics_geodetic*, which takes position and orientation in geodetic (latitude, longitude, altitude, and Euler angles with respect to ENU) coordinates. This command assumes zero velocity; therefore setting reference kinematics with the geodetic command while the radar is moving will greatly reduce radar performance.
- *get_reference_kinematics_ecef* returns the same parameters as the setter. Use this command to view the current ECEF reference frame kinematics.
- *get_reference_kinematics_geodetic* returns the same parameters as the setter. Use this command to view the current geodetic reference frame kinematics.

Like all of the kinematics setters and getters, these commands can be called at any time, but the kinematics must be set at least once before the radar can start transmitting. In addition, the radar-to-reference offsets must be set using *set_reference_frame_offsets* prior to setting reference kinematics.

If a user sets kinematics using *set_radar_kinematics*, it will leave the reference frame offsets unaffected, and will therefore update the position of *both* the radar frame and the reference frame.

11.3 Determining Radar-to-Reference Offsets

The radar-to-reference position and orientation offsets typically have to be determined by hand. Together, the six values describe the position and rotation of the radar frame relative to the reference frame.

Reference frame offsets can be set with the *set_reference_frame_offsets* command, and must be set at least once before the reference kinematics can be updated. They only need to be set once, and the offsets, like the kinematics themselves, will persist unless the radar is power cycled. After setting the offsets, the kinematics can be set any number of times and the same offsets will be applied to each update to derive the kinematics of the radar. The offsets are:

- *position_relative_to_reference_x*
- *position_relative_to_reference_y*
- *position_relative_to_reference_z*
- *rotation_relative_to_reference_yaw*
- *rotation_relative_to_reference_pitch*
- *rotation_relative_to_reference_roll*

Position_relative_to_reference_x, position_relative_to_reference_y, and *position_relative_to_reference_z* describe the linear position of the radar's origin (center of the front face of the radar) in the reference frame's x, y, and z coordinate axes.

The parameters *rotation_relative_to_reference_yaw, rotation_relative_to_reference_pitch,* and *rotation_relative_to_reference_roll* are the right-handed rotations (in degrees) about the radar's z-, y-, and x-axes as defined in *Echoshield Coordinate Frame Definitions*. To determine them, start with the radar reference frames aligned:

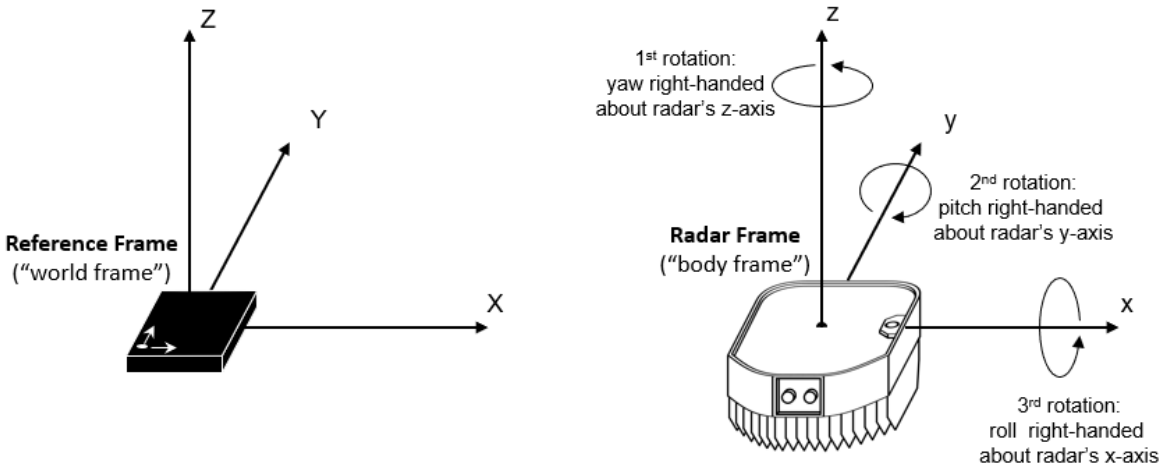


Fig. 7: Sequence of rotations used to specify the orientation of the Radar Frame relative to the Reference Frame. Conceptually, the radar starts with its axes aligned with the Reference Frame's axes, as shown. Then the radar is rotated about it's axes, one-at-a-time in the sequence shown, to get it into the final orientation relative to the Reference Frame. Note that there are multiple possible sequences of rotations that can be used to get to any final orientation, but most of the time only two are needed (for example a yaw rotation followed by a pitch rotation, with no roll rotation applied).

Then, rotate the radar away towards its desired orientation. First measure the yaw around the radar frame's z-axis. The order of this matters: EchoShield Euler angles are intrinsic, meaning that they are rotations around the radar frame's coordinate axes (as opposed to extrinsic, where the angles are around the stationary, or reference frame's, coordinate axes), so next measure pitch around the radar's *newly rotated* y-axis, then roll around the radar's new x-axis.

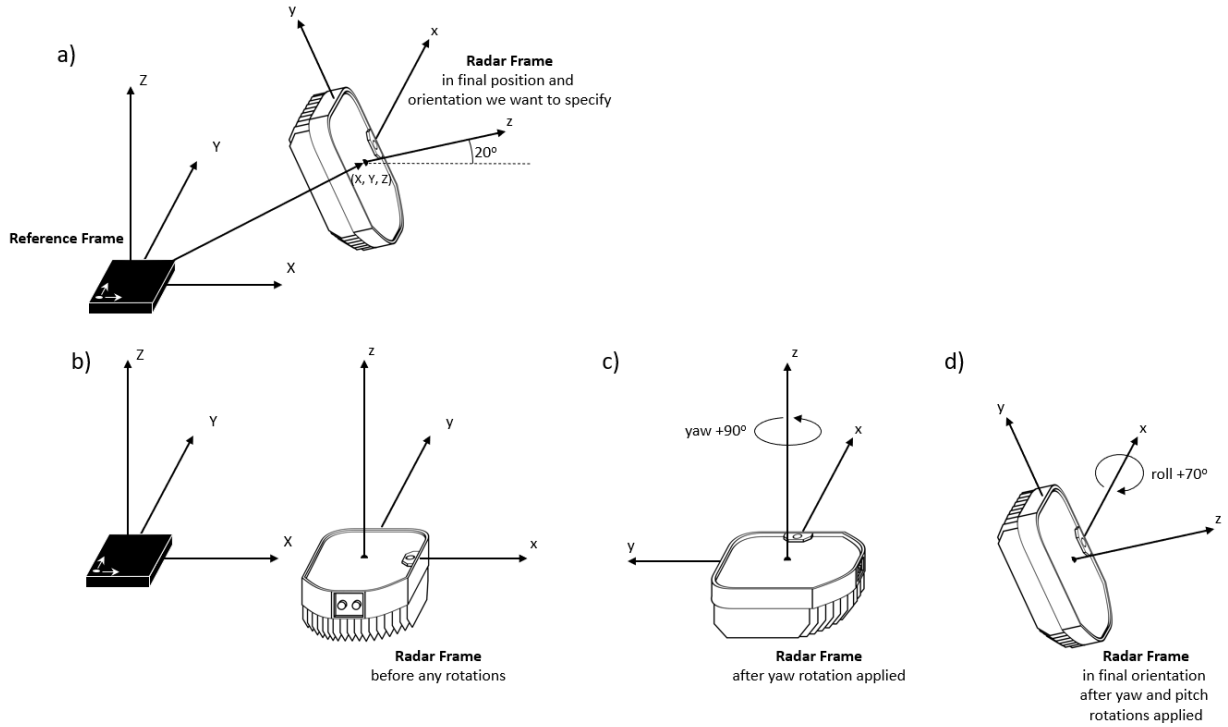


Fig. 8: Process to determine radar-to-reference offsets. a) The radar-to-reference relationship we want to specify. b) Start with the radar's axes parallel to the reference's axes, then c) determine the yaw rotation about the radar's z-axis needed to orient the radar to its final orientation. Then d) determine the intrinsic pitch arounnd radar's y-axis (zero in this example), then roll rotation about the radar's x-axis. After the rotations, the radar has traveled from being aligned to the reference frame to the desired orientation.

Note that there is not a unique sequence of yaw, pitch, and roll rotations for a given final rotation, and other values for these angles can achieve the same result.

After the offsets have been determined and a full set of kinematics data entered, a user should call *get_radar_kinematics* and *get_ins_data*. A large difference between the results of these commands (> 0.01 degrees latitude and longitude, > 2 meters altitude, > 2 degrees yaw, pitch and roll) implies an incorrect reference frame setup.

11.4 Example: Using a Sunsight AAT as the Reference Frame

A good example for setting up and measuring a reference frame uses the Sunsight™ Antenna Alignment Tool (AAT). Echodyne frequently uses the AAT to measure the location and orientation of EchoShield radars in stationary applications.

This section is meant to provide an example for setting up a reference frame. For steps to use the Sunsight AAT in a field deployment, please see the *Kinematics Setup Guide*.

The AAT reports location in standard geodetic latitude, longitude, and height-above-mean-sea-level (A.K.A. **orthometric**) altitude coordinates, but it reports orientation in a custom format.

The AAT measures orientation as three Euler angles in one of four different AAT mountings. This guide will use the **AAT faces back** setting. The other settings will rotate the reference frame and will require different conversions for heading, pitchback, and tilt. For this setting, the reported angles are:

Azimuth - The left-handed rotation about the AAT's z-axis (with zero azimuth when the AAT's x-axis points South). This is 180 degrees from the radar's heading when mounted using the bracket.

Tilt - The right-handed rotation about the AAT's y-axis with zero pitch when the AAT's x-axis is level (in the East-North plane).

Roll - The right-handed rotation about the AAT's x-axis with zero roll when the AAT's y-axis is level (in the East-North plane).

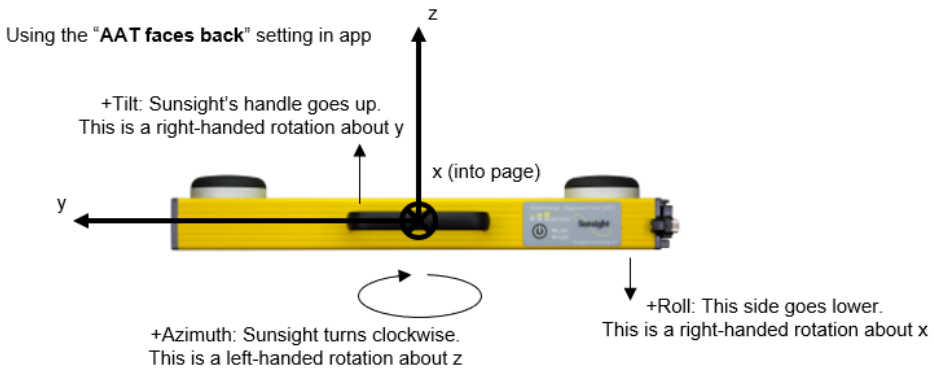


Fig. 9: Sunsight AAT azimuth, tilt, and roll directions and axes with the **AAT faces back** setting in app.

To use a Sunsight AAT as the reference frame for an EchoShield radar we need to convert these non-standard Euler angles to right-handed Euler angles relative to ENU.

The AAT's "azimuth" angle is positive clockwise (left-handed about Up) and is zero when the Sunsight's x-axis is pointed north. To make this a right-handed rotation we negate the sign of the azimuth value reported. Since zero yaw w.r.t. ENU should have the x-axis pointed parallel to East, we add 90^{o} to the sign-flipped azimuth. Hence, $\text{yaw}_{\text{ENU}} = 90 - \text{azimuth}_{\text{AAT}}$. The AAT's tilt angle is then the same as pitch, and the AAT's roll angle is roll.

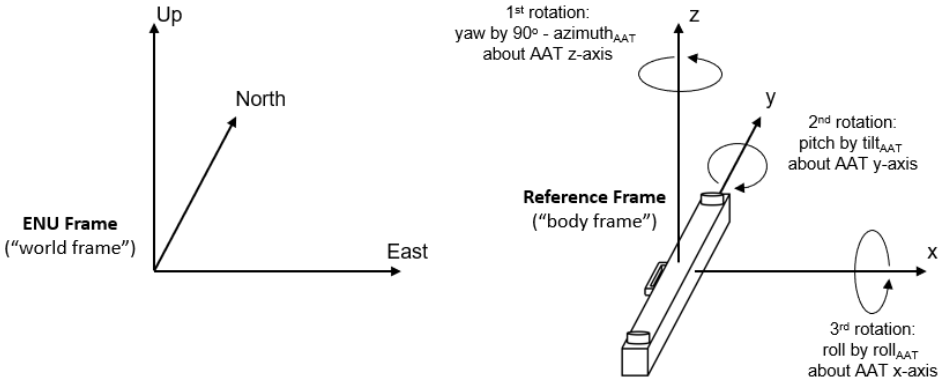


Fig. 10: Sunsight AAT geodetic YPR rotations for **AAT faces back** setting in app. The AAT is shown in an orientation of yaw = pitch = roll = 0^o ($\text{azimuth}_{\text{AAT}} = 90^o$, $\text{tilt}_{\text{AAT}} = 0^o$ $\text{roll}_{\text{AAT}} = 0^o$).

The relationships between the reference frame's geodetic YPR values and the AAT's reported Euler angles are then,

$$yaw_{ENU} = 90^o - azimuth_{AAT}$$

$$pitch_{ENU} = tilt_{AAT}$$

$$roll_{ENU} = roll_{AAT}$$

With the AAT's geodetic YPR values sorted out, we now need to determine the position- and rotation-relative-to-reference values.

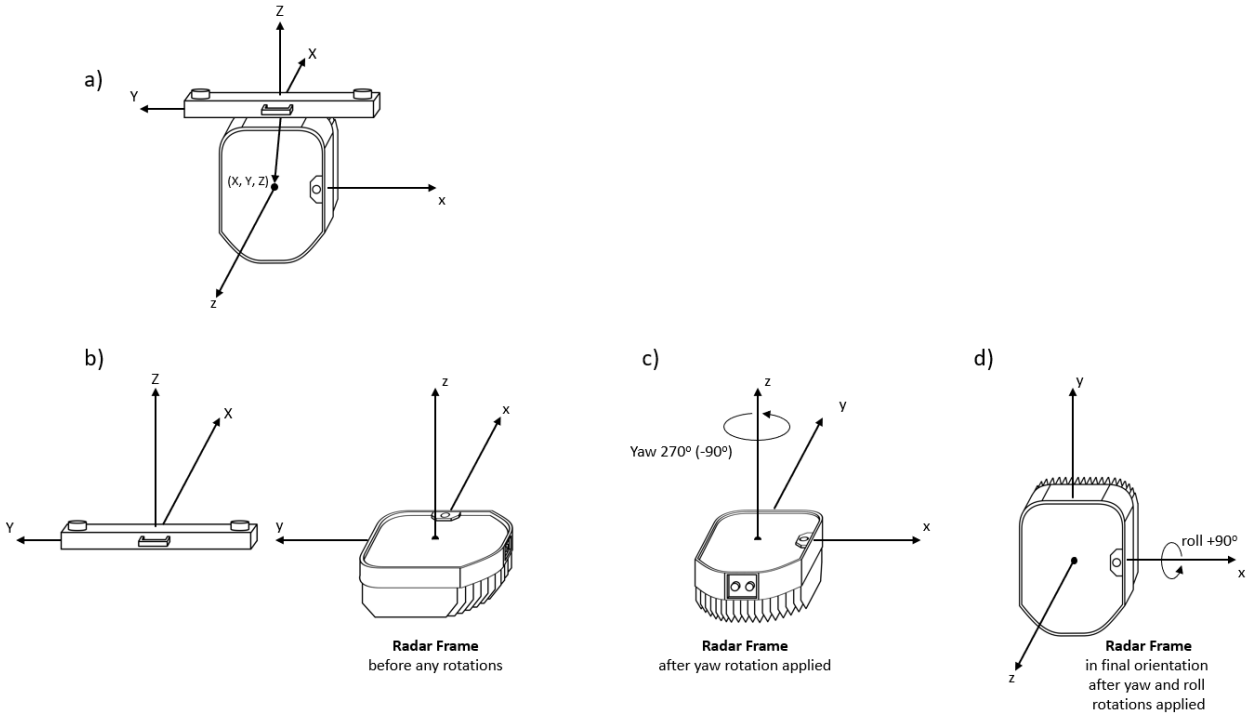


Fig. 11: Process to determine radar-to-reference offsets when using a Sunsight^{TM} bar as the reference. a) The radar-to-reference relationship we want to specify. From this mounting we measure the position of the center of the radar's face in the AAT's coordinate frame. To determine the rotations-relative-to-reference we b) start with the radar's axes parallel to the AAT's axes, then c) determine the intrinsic yaw rotation needed to orient the radar to its final orientation. Then starting from the orientation of the radar after it has been yawed, d) determine the intrinsic roll rotation needed to put the radar in its final orientation relative to the Reference Frame. We have applied zero pitch rotation.

Using the method from the previous section, we determine that the appropriate offsets are:

$$position\_relative\_to\_reference\_x = \ -0.035$$

$$position\_relative\_to\_reference\_y = 0$$

$$position\_relative\_to\_reference\_z = -\ 0.255$$

$$rotation\_relative\_to\_reference\_yaw = -90$$

$$rotation\_relative\_to\_reference\_pitch = 0$$

$$rotation\_relative\_to\_reference\_roll = 90$$

In summary,

1. Use *Sunsight faces back* setting in the Sunsight app.
2. Issue the *set_reference_frame_offsets* command with the following parameters:

- position_relative_to_reference_x = -0.035
- position_relative_to_reference_y = 0
- position_relative_to_reference_z = -0.255
- rotation_relative_to_reference_yaw = -90
- rotation_relative_to_reference_pitch = 0
- rotation_relative_to_reference_roll = 90

3. Issue the *set_reference_kinematics_geodetic* command with the following parameters:
   - latitude_deg = $\text{latitude}_{\text{AAT}}$
   - longitude_deg = $\text{longitude}_{\text{AAT}}$
   - altitude_m = $\text{altitude}_{\text{AAT}}$
   - yaw_enu_deg = 90 - $\text{azimuth}_{\text{AAT}}$
   - pitch_enu_deg = tilt_{AAT}
   - roll_enu_deg = roll_{AAT}
   - altitude_type = orthometric

12 Internal INS Usage

12.1 Introduction

Echoshield units are manufactured with an internal Inertial Navigation System (INS) which filters measurements from the onboard IMU and GPS sensors, which can serve as a basic estimate of radar kinematics. This chapter contains information on the features of the internal INS and defines its intended use cases.

12.2 Intended Use Cases

The internal INS is designed to operate under the following assumptions:

- Radar is stationary during use.
- GPS environment is ideal.

Since the filters in the internal INS are designed with the stationary case in mind, if the radar is moved while powered on, the measurements taken from this movement will be ingested into the internal filters. If this happens, the user should wait for 60 seconds before using the INS estimate. After that period, if all other conditions of operation are met, the INS can be used as an estimate of radar kinematics.

A user can directly input the data from the *get_ins_data* output into the *set_radar_kinematics* command to set kinematics; however, the internal estimate of heading will always have relatively high uncertainty, and

it is recommended for a user to use an external measurement of heading when possible. See the "Expected Uncertainties" section below for more information on the uncertainties of the internal INS outputs.

12.3 Kinematics Output

An up-to-date INS solution can be accessed with the *get_ins_data* command. The command returns the radar's internal estimate of kinematics:

- **latitude_deg**, which is calculated from the internal GPS.
- **longitude_deg**, which is calculated from the internal GPS.
- **altitude_deg**, which is calculated from the internal GPS.
- **heading_deg**, which is calculated from the internal GPS and filtered using an exponential decay filter.
- **pitchback_deg**, which is calculated from the internal IMU and filtered using an exponential decay filter.
- **tilt_deg**, which is calculated from the internal IMU and filtered using an exponential decay filter.
- **quaternion_ecef_[w, x, y, z]**, which is a quaternion representation of the INS frame rotation w.r.t. ECEF. This is derived from the heading, pitchback and tilt solutions.

For detailed definitions of the output data, see *Echoshield Coordinate Frame Definitions*.

The output also contains an **altitude_type** output parameter, which defines the output datum for the altitude solution. The optional **altitude_type** input lets a user toggle the desired altitude output datum, which can be either orthometric or ellipsoidal. See *Echoshield Coordinate Frame Definitions* for a detailed description of the difference between ellipsoidal and orthometric altitude datums.

12.4 Solution Validity

The INS data also includes additional information about the solution. Before using the INS solution, a user should refer to these parameters to ensure their validity:

- **pitch_roll_mature** describes the stability of the solution of the gravity-relative parameters pitchback and tilt. It is the only source of validity for those two parameters.
- **heading_mature** is true if the INS has been filtering the heading_deg solution for long enough to have a stable solution.
- **gps_fix_mode** is the satellite lock mode from the onboard GPS module. If it is NO LOCK, all of the GPS-derived parameters should be considered invalid. If it is 2D, the altitude and heading should be considered invalid. If it is 3D lock, the GPS module has a complete lock on a satellite constellation.
- **gps_heading_valid** is an additional flag from the GPS module describing the validity of the internal heading solution. If it is invalid, the internal heading solution should be considered invalid.

Since the INS pitchback and tilt are derived only from the IMU sensor, its solution can be considered valid if the **pitch_roll_mature** flag is valid, regardless of the state of the GPS sensor. The INS will update those values separately from the GPS-derived values.

If any of these flags are false, their respective output parameter should be considered unreliable; for instance, if the **gps_heading_valid** flag is true, but **heading_mature** is false, the internal heading solution should be considered invalid.

12.5 Accelerometer Calibration

Newly-manufactured radars may have accelerometer calibration coefficients, which will improve the pitchback_deg and tilt_deg solutions. The presence of the enhanced calibration coefficients is determined by the **enhanced_accelerometer_cal** parameter in the *get_ins_data* command and the *get_idn* command.

If enhanced_accelerometer_cal is false, the IMU-derived output will be less accurate. In this configuration, the **pitchback_deg** and **tilt_deg** solutions will have higher expected error and should not be used as a direct source of radar kinematics.

The GPS-derived parameters (latitude_deg, longitude_deg, altitude_m, and heading_deg) are unaffected by the accelerometer calibration.

12.6 Expected Uncertainties in INS Solution

The following is the expected INS state errors of an Echoshield unit with enhanced accelerometer calibration and valid states:

Table 237: INS Solution Uncertainties

Parameter	RMS Error	Unit
Horizontal Position	2.2	meters
Altitude	4.0	meters
Heading	0.93	degrees
Pitchback	0.48	degrees
Tilt	0.48	degrees