

---

# **Livox LiDAR SDK C/C++ API Reference**

**Livox**

**Jan 17, 2019**



## CONTENTS:

<b>1</b>	<b>Basic Types and Functions</b>	<b>1</b>
<b>2</b>	<b>General Functions</b>	<b>7</b>
2.1	Query Device Information . . . . .	7
2.2	Receive Point Cloud Data . . . . .	7
2.3	Set Coordinate System . . . . .	8
2.4	Error Message From Device . . . . .	9
2.5	Configure Static/Dynamic IP . . . . .	9
<b>3</b>	<b>Livox Hub Functions</b>	<b>11</b>
3.1	Query Connected LiDAR Unit Information . . . . .	11
3.2	Configure Lidars Mode . . . . .	12
3.3	Sampling Control . . . . .	14
3.4	Slot Power Control . . . . .	14
3.5	Configure Livox Hub Extrinsic Parameters . . . . .	15
3.6	Enable Lidar Unit Calculating Extrinsic Parameters . . . . .	18
3.7	Enable or Disable The Rain and Fog Mode. . . . .	18
<b>4</b>	<b>LiDAR Functions</b>	<b>21</b>
4.1	Configure LiDAR Mode . . . . .	21
4.2	Sample Control . . . . .	21
4.3	Configure LiDAR Extrinsic Parameters . . . . .	22
4.4	Enable and Disable the Rain/Fog Suppression . . . . .	23
	<b>Index</b>	<b>25</b>



## BASIC TYPES AND FUNCTIONS

### **enum DeviceType**

Device type.

*Values:*

**kDeviceTypeHub** = 0

Livox Hub.

**kDeviceTypeLidarMid40** = 1

Mid-40.

**kDeviceTypeLidarTele** = 2

Tele.

**kDeviceTypeLidarHorizon** = 3

Horizon.

### **enum LidarState**

Lidar state.

*Values:*

**kLidarStateInit** = 0

Initialization state.

**kLidarStateNormal** = 1

Normal work state.

**kLidarStatePowerSaving** = 2

Power-saving state.

**kLidarStateStandBy** = 3

Standby state.

**kLidarStateError** = 4

Error state.

**kLidarStateUnknown** = 5

### **enum LidarFeature**

Lidar feature.

*Values:*

**kLidarFeatureNone** = 0

No feature.

**kLidarFeatureRainFog** = 1

Rain and fog feature.

### enum ResponseStatus

Function return value definition.

*Values:*

**kStatusSuccess** = 0

Success.

**kStatusFailure** = 1

Failure.

**kStatusNotConnected** = 2

Requested device is not connected.

**kStatusNotSupported** = 3

Operation is not supported on this device.

**kStatusTimeout** = 4

Operation timeouts.

**kStatusNotEnoughMemory** = 5

No enough memory.

### enum DeviceEvent

Device update type, indicating the change of device connection or working state.

*Values:*

**kEventConnect** = 0

Device is connected.

**kEventDisconnect** = 1

Device is removed.

**kEventStateChange** = 2

Device working state changes or an error occurs.

### enum TimestampType

Timestamp sync mode define.

*Values:*

**kTimestampTypeNoSync** = 0

No sync signal mode.

**kTimestampTypePtp** = 1

1588v2.0 PTP sync mode.

**kTimestampTypeRsvd** = 2

Reserved use.

**kTimestampTypePpsGps** = 3

pps+gps sync mode.

**kTimestampTypePps** = 4

pps only sync mode.

**kTimestampTypeUnknown** = 5

Unknown mode.

### struct DeviceInfo

Information of the connected LiDAR or hub.

**Public Members**

char **broadcast\_code**[16]  
Device broadcast code, null-terminated string, 15 characters at most.

uint8\_t **handle**  
Device handle.

uint8\_t **slot**  
Slot number used for connecting LiDAR.

uint8\_t **id**  
LiDAR id.

uint32\_t **type**  
Device type, refer to *DeviceType*.

uint16\_t **data\_port**  
Point cloud data UDP port.

uint16\_t **cmd\_port**  
Control command UDP port.

char **ip**[16]  
IP address.

*LidarState* **state**  
LiDAR state.

*LidarFeature* **feature**  
LiDAR feature.

*StatusUnion* **status**

**union StatusUnion**  
*#include <livox\_def.h>* Information of LiDAR work state.

**Public Members**

uint32\_t **status\_code**  
LiDAR work state status code.

uint32\_t **progress**  
LiDAR work state switching progress.

bool **Init** ()  
Initialize the SDK.  
**Return** true if successfully initialized, otherwise false.

bool **Start** ()  
Start the device scanning routine which runs on a separate thread.  
**Return** true if successfully started, otherwise false.

void **Uninit** ()  
Uninitialize the SDK.

**struct BroadcastDeviceInfo**  
The information of broadcast device.

### Public Members

char **broadcast\_code**[16]

Device broadcast code, null-terminated string, 15 characters at most.

uint8\_t **dev\_type**

Device type, refer to *DeviceType*.

uint16\_t **reserved**

Reserved.

**typedef** void (\***DeviceBroadcastCallback**) (const *BroadcastDeviceInfo* \*info)

SetBroadcastCallback response callback function.

### Parameters

- info: information of the broadcast device, becomes invalid after the function returns.

void **SetBroadcastCallback** (*DeviceBroadcastCallback* cb)

Set the callback of listening device broadcast message. When broadcast message is received from Livox Hub/LiDAR, cb is called.

### Parameters

- cb: callback for device broadcast.

**typedef** void (\***DeviceStateUpdateCallback**) (const *DeviceInfo* \*device, *DeviceEvent* type)

SetDeviceStateUpdateCallback response callback function.

### Parameters

- device: information of the connected device.
- type: the update type that indicates connection/disconnection of the device or change of working state.

void **SetDeviceStateUpdateCallback** (*DeviceStateUpdateCallback* cb)

Add a callback for device connection or working state changing event.

**Note** Livox SDK supports two hardware connection modes. 1: Directly connecting to the LiDAR device; 2. Connecting to the LiDAR device(s) via the Livox Hub. In the first mode, connection/disconnection of every LiDAR unit is reported by this callback. In the second mode, only connection/disconnection of the Livox Hub is reported by this callback. If you want to get information of the LiDAR unit(s) connected to hub, see *HubQueryLidarInformation*.

**Note** 3 conditions can trigger this callback:

1. Connection and disconnection of device.
2. A change of device working state.
3. An error occurs.

### Parameters

- cb: callback for device connection/disconnection.

uint8\_t **AddHubToConnect** (const char \*broadcast\_code, uint8\_t \*handle)

Add a broadcast code to the connecting list and only devices with broadcast code in this list will be connected. The broadcast code is unique for every device.

**Return** kStatusSuccess on successful return, see *ResponseStatus* for other error code.

### Parameters



- `broadcast_code`: device's broadcast code.
- `handle`: device handle. For Livox Hub, the handle is always 31; for LiDAR units connected to the Livox Hub, the corresponding handle is  $(\text{slot}-1)*3+\text{id}-1$ .

`uint8_t AddLidarToConnect` (`const char *broadcast_code`, `uint8_t *handle`)

Add a broadcast code to the connecting list and only devices with broadcast code in this list will be connected. The broadcast code is unique for every device.

**Return** `kStatusSuccess` on successful return, see [ResponseStatus](#) for other error code.

**Parameters**

- `broadcast_code`: device's broadcast code.
- `handle`: device handle. The handle is the same as the order calling `AddLidarToConnect` starting from 0.

`uint8_t GetConnectedDevices` (`DeviceInfo *devices`, `uint8_t *size`)

Get all connected devices' information.

**Return** `kStatusSuccess` on successful return, see [ResponseStatus](#) for other error code.

**Parameters**

- `devices`: list of connected devices' information.
- `size`: number of devices connected.



## GENERAL FUNCTIONS

### 2.1 Query Device Information

**struct DeviceInformationResponse**  
The response body of querying device information.

#### Public Members

`uint8_t ret_code`  
Return code.

`uint8_t firmware_version[4]`  
Firmware version.

**typedef void (\*DeviceInformationCallback)** (`uint8_t status`, `uint8_t handle`, *DeviceInformationResponse* \*response, `void *client_data`)  
Function type of callback that queries device's information.

#### Parameters

- `status`: `kStatusSuccess` on successful return, `kStatusTimeout` on timeout, see *ResponseStatus* for other error code.
- `handle`: device handle.
- `response`: response from the device.
- `client_data`: user data associated with the command.

`uint8_t QueryDeviceInformation` (`uint8_t handle`, *DeviceInformationCallback* `cb`, `void *client_data`)  
Command to query device's information.

**Return** `kStatusSuccess` on successful return, see *ResponseStatus* for other error code.

#### Parameters

- `handle`: device handle.
- `cb`: callback for the command.
- `client_data`: user data associated with the command.

### 2.2 Receive Point Cloud Data

**struct LivoxEthPacket**  
Point cloud packet.

### Public Members

`uint8_t version`

Packet protocol version.

`uint8_t slot`

Slot number used for connecting LiDAR.

`uint8_t id`

LiDAR id.

`uint8_t rsvd`

Reserved.

`uint32_t err_code`

Device error status indicator information.

`uint8_t timestamp_type`

Timestamp type.

`uint8_t data_type`

Point cloud coordinate format, 1 for spherical coordinate data, 0 for cartesian coordinate data.

`uint8_t timestamp[8]`

Nanosecond or UTC format timestamp.

`uint8_t data[1]`

Point cloud data.

**typedef** void (\*DataCallback) (uint8\_t handle, *LivoxEthPacket* \*data, uint32\_t data\_num)  
Callback function for receiving point cloud data.

#### Parameters

- handle: device handle.
- data: device's data.
- data\_num: number of points in data.

void **SetDataCallback** (uint8\_t handle, *DataCallback* cb)

Set the callback to receive point cloud data. Only one callback is supported for a specific device. Set the point cloud data callback before beginning sampling.

#### Parameters

- cb: callback to receive point cloud data.

`uint8_t HubGetLidarHandle` (uint8\_t slot, uint8\_t id)

Get the LiDAR unit handle used in the Livox Hub data callback function from slot and id.

**Return** LiDAR unit handle.

#### Parameters

- slot: Livox Hub's slot.
- id: Livox Hub's id.

## 2.3 Set Coordinate System

`uint8_t SetCartesianCoordinate` (uint8\_t handle, *CommonCommandCallback* cb, void \*client\_data)

Command to change point cloud coordinate system to cartesian coordinate.

**Return** kStatusSuccess on successful return, see *ResponseStatus* for other error code.

**Parameters**

- `handle`: device handle.
- `cb`: callback for the command.
- `client_data`: user data associated with the command.

`uint8_t SetSphericalCoordinate` (`uint8_t handle`, *CommonCommandCallback* `cb`, `void *client_data`)  
Change point cloud coordinate system to spherical coordinate.

**Return** `kStatusSuccess` on successful return, see *ResponseStatus* for other error code.

**Parameters**

- `handle`: device handle.
- `cb`: callback for the command.
- `client_data`: user data associated with the command.

## 2.4 Error Message From Device

**struct ErrorMessage**

The response body of getting device error status.

**Public Members**

`uint32_t error_code`  
Error code.

**typedef** `void (*ErrorMessageCallback)` (`uint8_t handle`, *ErrorMessage* `*message`)  
Callback of the error status message.

**Parameters**

- `handle`: device handle.
- `response`: response from the device.

`uint8_t SetErrorMessageCallback` (`uint8_t handle`, *ErrorMessageCallback* `cb`)  
Add error status callback for the device.

**Return** `kStatusSuccess` on successful return, see *ResponseStatus* for other error code.

**Parameters**

- `handle`: device handle.
- `cb`: callback for the command.

## 2.5 Configure Static/Dynamic IP

**struct SetDeviceIPModeRequest**

The request body of the command for setting device's IP mode.

**Public Members**

`uint8_t ip_mode`  
IP address mode: 0 for dynamic IP address, 1 for static IP address.

`uint32_t ip_addr`  
IP address.

`uint8_t SetStaticDynamicIP (uint8_t handle, SetDeviceIPModeRequest *req, CommonCommandCallback cb, void *client_data)`

Set device's IP mode.

**Return** `kStatusSuccess` on successful return, see *ResponseStatus* for other error code.

### Parameters

- `handle`: device handle.
- `req`: request sent to device.
- `cb`: callback for the command.
- `client_data`: user data associated with the command.

**struct GetDeviceIPModeResponse**

The response body of getting device's IP mode.

### Public Members

`uint8_t ret_code`

Return code.

`uint8_t ip_mode`

IP address mode: 0 for dynamic IP address, 1 for static IP address.

`uint32_t ip_addr`

IP address.

**typedef void (\*GetDeviceIPInformationCallback) (uint8\_t status, uint8\_t handle, GetDeviceIPModeResponse \*response, void \*client\_data)**

Callback function that gets device's IP information.

### Parameters

- `status`: `kStatusSuccess` on successful return, `kStatusTimeout` on timeout, see *ResponseStatus* for other error code.
- `handle`: device handle.
- `response`: response from the device.
- `client_data`: user data associated with the command.

`uint8_t GetDeviceIPInformation (uint8_t handle, GetDeviceIPInformationCallback cb, void *client_data)`

Get device's IP mode.

**Return** `kStatusSuccess` on successful return, see *ResponseStatus* for other error code.

### Parameters

- `handle`: device handle.
- `cb`: callback for the command.
- `client_data`: user data associated with the command.

## LIVOX HUB FUNCTIONS

### 3.1 Query Connected LiDAR Unit Information

#### **struct ConnectedLidarInfo**

The information of LiDAR units that are connected to the Livox Hub.

##### **Public Members**

char **broadcast\_code**[16]

Device broadcast code, null-terminated string, 15 characters at most.

uint8\_t **dev\_type**

Device type, refer to *DeviceType*.

uint8\_t **version**[4]

Firmware version.

uint8\_t **slot**

Slot number used for connecting LiDAR units.

uint8\_t **id**

Device id.

#### **struct HubQueryLidarInformationResponse**

The response body of querying the information of LiDAR units connected to the Livox Hub.

##### **Public Members**

uint8\_t **ret\_code**

Return code from device.

uint8\_t **count**

Count of device\_info\_list.

*ConnectedLidarInfo* **device\_info\_list**[1]

Connected lidars information.

**typedef** void (**\*HubQueryLidarInformationCallback**) (uint8\_t status, uint8\_t handle, *HubQueryLidarInformationResponse* \*response, void \*client\_data)

HubQueryLidarInformation response callback function.

##### **Parameters**

- status: kStatusSuccess on successful return, kStatusTimeout on timeout, see *ResponseStatus* for other error code.
- handle: device handle.

- `response`: response from the device.
- `client_data`: user data associated with the command.

`uint8_t HubQueryLidarInformation` (*HubQueryLidarInformationCallback* `cb`, `void *client_data`)

Query the information of LiDARs connected to the hub.

**Return** `kStatusSuccess` on successful return, see *ResponseStatus* for other error code.

#### Parameters

- `cb`: callback for the command.
- `client_data`: user data associated with the command.

## 3.2 Configure Lidars Mode

### **struct HubSetModeResponse**

The response of setting LiDAR units working mode.

#### Public Members

`uint8_t ret_code`

Return code from device.

`uint8_t count`

Count of `ret_state_list`.

`ReturnCode ret_state_list[1]`

Return status list.

**typedef** `void (*HubSetModeCallback)` (`uint8_t status`, `uint8_t handle`, *HubSetModeResponse* `*response`, `void *client_data`)

HubSetMode response callback function.

#### Parameters

- `status`: `kStatusSuccess` on successful return, `kStatusTimeout` on timeout, see *ResponseStatus* for other error code.
- `handle`: device handle.
- `response`: response from the device.
- `client_data`: user data associated with the command.

### **struct HubSetModeRequest**

The request body of setting LiDAR units working mode.

#### Public Members

`uint8_t count`

Number of LiDAR units to set.

*LidarModeRequestItem* `config_list[1]`

LiDAR mode configuration list.

### **struct LidarModeRequestItem**

LiDAR mode configuration information.



**Public Members**

char **broadcast\_code**[16]

Device broadcast code, null-terminated string, 15 characters at most.

uint8\_t **state**

LiDAR state.

uint8\_t **HubSetMode** (*HubSetModeRequest* \*req, uint16\_t length, *HubSetModeCallback* cb, void \*client\_data)

Set the mode of LiDAR unit connected to the Livox Hub.

**Return** kStatusSuccess on successful return, see *ResponseStatus* for other error code.

**Parameters**

- req: mode configuration of LiDAR units.
- length: length of req.
- cb: callback for the command.
- client\_data: user data associated with the command.

**struct LidarStateItem**

**Public Members**

char **broadcast\_code**[16]

Broadcast code.

uint8\_t **state**

LiDAR state.

uint8\_t **feature**

LiDAR feature.

*StatusUnion* **error\_union**

**struct HubQueryLidarStatusResponse**

The response body of getting sub LiDAR's state.

**Public Members**

uint8\_t **ret\_code**

Return code.

uint8\_t **count**

Number of LiDAR connected to the Livox Hub.

*LidarStateItem* **state\_list**[1]

Device information of connected LiDAR units.

**typedef** void (\***HubQueryLidarStatusCallback**) (uint8\_t status, uint8\_t handle, *HubQueryLidarStatusResponse* \*response, void \*client\_data)

HubQueryLidarStatus response callback function.

**Parameters**

- status: kStatusSuccess on successful return, kStatusTimeout on timeout, see *ResponseStatus* for other error code.
- handle: device handle.
- response: response from the device.

- `client_data`: user data associated with the command.

`uint8_t HubQueryLidarStatus` (*HubQueryLidarStatusCallback* `cb`, `void *client_data`)  
Get the state of LiDAR units connected to the Livox Hub.

**Return** `kStatusSuccess` on successful return, see *ResponseStatus* for other error code.

**Parameters**

- `cb`: callback for the command.
- `client_data`: user data associated with the command.

### 3.3 Sampling Control

`typedef void (*CommonCommandCallback)` (`uint8_t status`, `uint8_t handle`, `uint8_t response`, `void *client_data`)  
Function type of callback with 1 byte of response.

**Parameters**

- `status`: `kStatusSuccess` on successful return, `kStatusTimeout` on timeout, see *ResponseStatus* for other error code.
- `handle`: device handle.
- `response`: response from the device.
- `client_data`: user data associated with the command.

`uint8_t HubStartSampling` (*CommonCommandCallback* `cb`, `void *client_data`)  
Start hub sampling.

**Return** `kStatusSuccess` on successful return, see *ResponseStatus* for other error code.

**Parameters**

- `cb`: callback for the command.
- `client_data`: user data associated with the command.

`uint8_t HubStopSampling` (*CommonCommandCallback* `cb`, `void *client_data`)  
Stop the Livox Hub's sampling.

**Return** `kStatusSuccess` on successful return, see *ResponseStatus* for other error code.

**Parameters**

- `cb`: callback for the command.
- `client_data`: user data associated with the command.

### 3.4 Slot Power Control

`struct HubControlSlotPowerRequest`  
The request body of toggling the power supply of the slot.

**Public Members**

`uint8_t slot`  
Slot of the hub.

`uint8_t state`  
Status of toggling the power supply.

`uint8_t HubControlSlotPower` (*HubControlSlotPowerRequest* \*req, *CommonCommandCallback* cb, void \*client\_data)

Toggle the power supply of designated slots.

**Return** `kStatusSuccess` on successful return, see *ResponseStatus* for other error code.

#### Parameters

- req: request whether to enable or disable the power of designated slots.
- cb: callback for the command.
- client\_data: user data associated with the command.

## 3.5 Configure Livox Hub Extrinsic Parameters

**struct HubSetExtrinsicParameterResponse**

The response body of setting the Livox Hub's parameters.

#### Public Members

`uint8_t ret_code`  
Return code.

`uint8_t count`  
Count of `ret_code_list`.

`ReturnCode ret_code_list[1]`  
Return code

**typedef void (\*HubSetExtrinsicParameterCallback)** (`uint8_t` status, `uint8_t` handle, *HubSetExtrinsicParameterResponse* \*response, void \*client\_data)

`HubSetExtrinsicParameter` response callback function.

#### Parameters

- status: `kStatusSuccess` on successful return, `kStatusTimeout` on timeout, see *ResponseStatus* for other error code.
- handle: device handle.
- response: response from the device.
- client\_data: user data associated with the command.

**struct HubSetExtrinsicParameterRequest**

The request body of setting the Livox Hub's parameters.

#### Public Members

`uint8_t count`  
Count of `cfg_param_list`.

*ExtrinsicParameterRequestItem* `parameter_list[1]`  
Configuration parameter list.

**struct ExtrinsicParameterRequestItem**

LiDAR configuration information.

### Public Members

char **broadcast\_code**[16]  
Device broadcast code.

float **roll**  
Roll angle, unit: degree.

float **pitch**  
Pitch angle, unit: degree.

float **yaw**  
Yaw angle, unit: degree.

int32\_t **x**  
X translation, unit: mm.

int32\_t **y**  
Y translation, unit: mm.

int32\_t **z**  
Z translation, unit: mm.

uint8\_t **HubSetExtrinsicParameter** (*HubSetExtrinsicParameterRequest \*req*, uint16\_t *length*, *HubSetExtrinsicParameterCallback cb*, void \**client\_data*)  
Set extrinsic parameters of LiDAR units connected to the Livox Hub.

**Return** kStatusSuccess on successful return, see *ResponseStatus* for other error code.

### Parameters

- *req*: the parameters to write.
- *length*: the request's length.
- *cb*: callback for the command.
- *client\_data*: user data associated with the command.

**struct HubGetExtrinsicParameterRequest**  
The request body of getting the Livox Hub's parameters.

### Public Members

uint8\_t **count**  
Count of *code\_list*.

*DeviceBroadcastCode* **code\_list**[1]  
Broadcast code list.

**struct DeviceBroadcastCode**

### Public Members

char **broadcast\_code**[16]  
Device broadcast code.

**struct HubGetExtrinsicParameterResponse**  
The response body of getting the Livox Hub's parameters.

**Public Members**

uint8\_t **ret\_code**  
Return code.

uint8\_t **count**  
Number of LiDAR units connected to the Livox Hub.

*ExtrinsicParameterResponseItem* **parameter\_list**[1]  
Position parameters of connected LiDAR unit(s).

**struct ExtrinsicParameterResponseItem**  
LiDAR extrinsic parameters.

**Public Members**

uint8\_t **ret\_code**  
Return code.

char **broadcast\_code**[16]  
Broadcast code.

float **roll**  
Roll angle, unit: degree.

float **pitch**  
Pitch angle, unit: degree.

float **yaw**  
Yaw angle, unit: degree.

int32\_t **x**  
X translation, unit: mm.

int32\_t **y**  
Y translation, unit: mm.

int32\_t **z**  
Z translation, unit: mm.

**typedef void (\*HubGetExtrinsicParameterCallback)** (uint8\_t status, uint8\_t handle, *HubGetExtrinsicParameterResponse* \*response, void \*client\_data)

HubGetExtrinsicParameter response callback function.

**Parameters**

- status: kStatusSuccess on successful return, kStatusTimeout on timeout, see *ResponseStatus* for other error code.
- handle: device handle.
- response: response from the device.
- client\_data: user data associated with the command.

uint8\_t **HubGetExtrinsicParameter** (*HubGetExtrinsicParameterRequest* \*req, uint16\_t length, *HubGetExtrinsicParameterCallback* cb, void \*client\_data)

Get extrinsic parameters of LiDAR units connected to the Livox Hub.

**Return** kStatusSuccess on successful return, see *ResponseStatus* for other error code.

**Parameters**

- req: the LiDAR units broadcast code list.

- `length`: the request's length.
- `cb`: callback for the command.
- `client_data`: user data associated with the command.

### 3.6 Enable Lidar Unit Calculating Extrinsic Parameters

`uint8_t HubExtrinsicParameterCalculation` (bool *enable*, *CommonCommandCallback* *cb*, void *\*client\_data*)

Turn on or off the calculation of extrinsic parameters.

**Return** `kStatusSuccess` on successful return, see *ResponseStatus* for other error code.

#### Parameters

- `enable`: the request whether enable or disable calculating the extrinsic parameters.
- `cb`: callback for the command.
- `client_data`: user data associated with the command.

### 3.7 Enable or Disable The Rain and Fog Mode.

`struct RainFogSuppressRequestItem`

#### Public Members

char `broadcast_code`[16]  
Device broadcast code.

uint8\_t `feature`  
Close or open the rain and fog feature.

`struct HubRainFogSuppressRequest`

The request body of toggling the LiDAR units rain and fog mode.

#### Public Members

uint8\_t `count`  
Number of LiDAR units connected to the Livox Hub.

*RainFogSuppressRequestItem* `lidar_cfg_list`[1]  
Command data of connected LiDAR units.

`struct HubRainFogSuppressResponse`

The response body of toggling the LiDAR units rain and fog mode.

#### Public Members

uint8\_t `ret_code`  
Return code.

uint8\_t `count`  
Count of `ret_state_list`.

ReturnCode `ret_state_list`[1]  
Return code

---

**typedef** void (\***HubRainFogSuppressCallback**) (uint8\_t status, uint8\_t handle, *HubRainFogSuppressResponse* \*response, void \*client\_data)  
HubRainFogSuppress response callback function.

**Parameters**

- status: kStatusSuccess on successful return, kStatusTimeout on timeout, see *ResponseStatus* for other error code.
- handle: device handle.
- response: response from the device.
- client\_data: user data associated with the command.

uint8\_t **HubRainFogSuppress** (*HubRainFogSuppressRequest* \*req, uint16\_t length, *HubRainFogSuppressCallback* cb, void \*client\_data)  
Toggling the rain and fog mode for lidars connected to the hub.

**Return** kStatusSuccess on successful return, see *ResponseStatus* for other error code.

**Parameters**

- req: the request whether open or close the rain and fog mode.
- length: the request's length.
- cb: callback for the command.
- client\_data: user data associated with the command.





## LIDAR FUNCTIONS

### 4.1 Configure LiDAR Mode

**enum LidarMode**

Lidar mode.

*Values:*

**kLidarModeNormal** = 1

Normal mode.

**kLidarModePowerSaving** = 2

Power-saving mode.

**kLidarModeStandby** = 3

Standby mode.

**uint8\_t LidarSetMode** (uint8\_t *handle*, *LidarMode mode*, *CommonCommandCallback cb*, void  
\**client\_data*)

Set LiDAR mode.

**Note** Successful callback function status only means LiDAR successfully starting the changing process of mode.  
You need to observe the actually change of mode in *DeviceStateUpdateCallback* function.

**Return** *kStatusSuccess* on successful return, see *ResponseStatus* for other error code.

**Parameters**

- *handle*: device handle.
- *mode*: the mode to change.
- *cb*: callback for the command.
- *client\_data*: user data associated with the command.

### 4.2 Sample Control

**uint8\_t LidarStartSampling** (uint8\_t *handle*, *CommonCommandCallback cb*, void \**client\_data*)

Start LiDAR sampling.

**Return** *kStatusSuccess* on successful return, see *ResponseStatus* for other error code.

**Parameters**

- *handle*: device handle.
- *cb*: callback for the command.
- *client\_data*: user data associated with the command.

`uint8_t LidarStopSampling` (`uint8_t handle`, *CommonCommandCallback cb*, `void *client_data`)  
Stop LiDAR sampling.

**Return** `kStatusSuccess` on successful return, see *ResponseStatus* for other error code.

### Parameters

- `handle`: device handle.
- `cb`: callback for the command.
- `client_data`: user data associated with the command.

## 4.3 Configure LiDAR Extrinsic Parameters

**struct LidarSetExtrinsicParameterRequest**

The request body for the command of setting LiDAR's parameters.

### Public Members

float **roll**  
Roll angle, unit: degree.

float **pitch**  
Pitch angle, unit: degree.

float **yaw**  
Yaw angle, unit: degree.

int32\_t **x**  
X translation, unit: mm.

int32\_t **y**  
Y translation, unit: mm.

int32\_t **z**  
Z translation, unit: mm.

`uint8_t LidarSetExtrinsicParameter` (`uint8_t handle`, *LidarSetExtrinsicParameterRequest \*req*, *CommonCommandCallback cb*, `void *client_data`)

Set LiDAR extrinsic parameters.

**Return** `kStatusSuccess` on successful return, see *ResponseStatus* for other error code.

### Parameters

- `handle`: device handle.
- `param`: the parameters to write.
- `cb`: callback for the command.
- `client_data`: user data associated with the command.

**struct LidarGetExtrinsicParameterResponse**

The response body of getting LiDAR's parameters.

### Public Members

uint8\_t **ret\_code**

float **roll**  
Roll angle, unit: degree.

float **pitch**  
Pitch angle, unit: degree.

float **yaw**  
Yaw angle, unit: degree.

int32\_t **x**  
X translation, unit: mm.

int32\_t **y**  
Y translation, unit: mm.

int32\_t **z**  
Z translation, unit: mm.

**typedef** void (\***LidarGetExtrinsicParameterCallback**) (uint8\_t status, uint8\_t handle, *LidarGetExtrinsicParameterResponse* \*response, void \*client\_data)

LidarGetExtrinsicParameter response callback function.

#### Parameters

- status: kStatusSuccess on successful return, kStatusTimeout on timeout, see *ResponseStatus* for other error code.
- handle: device handle.
- response: response from the device.
- client\_data: user data associated with the command.

uint8\_t **LidarGetExtrinsicParameter** (uint8\_t handle, *LidarGetExtrinsicParameterCallback* cb, void \*client\_data)

Get LiDAR extrinsic parameters.

**Return** kStatusSuccess on successful return, see *ResponseStatus* for other error code.

#### Parameters

- handle: device handle.
- cb: callback for the command.
- client\_data: user data associated with the command.

## 4.4 Enable and Disable the Rain/Fog Suppression

uint8\_t **LidarRainFogSuppress** (uint8\_t handle, bool enable, *CommonCommandCallback* cb, void \*client\_data)

Enable and disable the rain/fog suppression.

**Return** kStatusSuccess on successful return, see *ResponseStatus* for other error code.

#### Parameters

- handle: device handle.
- cb: callback for the command.
- client\_data: user data associated with the command.



## INDEX

### A

AddHubToConnect (C++ *function*), 4  
AddLidarToConnect (C++ *function*), 5

### B

BroadcastDeviceInfo (C++ *class*), 3  
BroadcastDeviceInfo::broadcast\_code (C++ *member*), 4  
BroadcastDeviceInfo::dev\_type (C++ *member*), 4  
BroadcastDeviceInfo::reserved (C++ *member*), 4

### C

CommonCommandCallback (C++ *type*), 14  
ConnectedLidarInfo (C++ *class*), 11  
ConnectedLidarInfo::broadcast\_code (C++ *member*), 11  
ConnectedLidarInfo::dev\_type (C++ *member*), 11  
ConnectedLidarInfo::id (C++ *member*), 11  
ConnectedLidarInfo::slot (C++ *member*), 11  
ConnectedLidarInfo::version (C++ *member*), 11

### D

DataCallback (C++ *type*), 8  
DeviceBroadcastCallback (C++ *type*), 4  
DeviceBroadcastCode (C++ *class*), 16  
DeviceBroadcastCode::broadcast\_code (C++ *member*), 16  
DeviceEvent (C++ *type*), 2  
DeviceInfo (C++ *class*), 2  
DeviceInfo::broadcast\_code (C++ *member*), 3  
DeviceInfo::cmd\_port (C++ *member*), 3  
DeviceInfo::data\_port (C++ *member*), 3  
DeviceInfo::feature (C++ *member*), 3  
DeviceInfo::handle (C++ *member*), 3  
DeviceInfo::id (C++ *member*), 3  
DeviceInfo::ip (C++ *member*), 3  
DeviceInfo::slot (C++ *member*), 3  
DeviceInfo::state (C++ *member*), 3

DeviceInfo::status (C++ *member*), 3  
DeviceInfo::type (C++ *member*), 3  
DeviceInformationCallback (C++ *type*), 7  
DeviceInformationResponse (C++ *class*), 7  
DeviceInformationResponse::firmware\_version (C++ *member*), 7  
DeviceInformationResponse::ret\_code (C++ *member*), 7  
DeviceStateUpdateCallback (C++ *type*), 4  
DeviceType (C++ *type*), 1

### E

ErrorMessage (C++ *class*), 9  
ErrorMessage::error\_code (C++ *member*), 9  
ErrorMessageCallback (C++ *type*), 9  
ExtrinsicParameterRequestItem (C++ *class*), 15  
ExtrinsicParameterRequestItem::broadcast\_code (C++ *member*), 16  
ExtrinsicParameterRequestItem::pitch (C++ *member*), 16  
ExtrinsicParameterRequestItem::roll (C++ *member*), 16  
ExtrinsicParameterRequestItem::x (C++ *member*), 16  
ExtrinsicParameterRequestItem::y (C++ *member*), 16  
ExtrinsicParameterRequestItem::yaw (C++ *member*), 16  
ExtrinsicParameterRequestItem::z (C++ *member*), 16  
ExtrinsicParameterResponseItem (C++ *class*), 17  
ExtrinsicParameterResponseItem::broadcast\_code (C++ *member*), 17  
ExtrinsicParameterResponseItem::pitch (C++ *member*), 17  
ExtrinsicParameterResponseItem::ret\_code (C++ *member*), 17  
ExtrinsicParameterResponseItem::roll (C++ *member*), 17



HubSetModeResponse::ret\_code (C++ member), 12

HubSetModeResponse::ret\_state\_list (C++ member), 12

HubStartSampling (C++ function), 14

HubStopSampling (C++ function), 14

## I

Init (C++ function), 3

## K

kDeviceTypeHub (C++ enumerator), 1

kDeviceTypeLidarHorizon (C++ enumerator), 1

kDeviceTypeLidarMid40 (C++ enumerator), 1

kDeviceTypeLidarTele (C++ enumerator), 1

kEventConnect (C++ enumerator), 2

kEventDisconnect (C++ enumerator), 2

kEventStateChange (C++ enumerator), 2

kLidarFeatureNone (C++ enumerator), 1

kLidarFeatureRainFog (C++ enumerator), 1

kLidarModeNormal (C++ enumerator), 21

kLidarModePowerSaving (C++ enumerator), 21

kLidarModeStandby (C++ enumerator), 21

kLidarStateError (C++ enumerator), 1

kLidarStateInit (C++ enumerator), 1

kLidarStateNormal (C++ enumerator), 1

kLidarStatePowerSaving (C++ enumerator), 1

kLidarStateStandBy (C++ enumerator), 1

kLidarStateUnknown (C++ enumerator), 1

kStatusFailure (C++ enumerator), 2

kStatusNotConnected (C++ enumerator), 2

kStatusNotEnoughMemory (C++ enumerator), 2

kStatusNotSupported (C++ enumerator), 2

kStatusSuccess (C++ enumerator), 2

kStatusTimeout (C++ enumerator), 2

kTimestampTypeNoSync (C++ enumerator), 2

kTimestampTypePps (C++ enumerator), 2

kTimestampTypePpsGps (C++ enumerator), 2

kTimestampTypePtp (C++ enumerator), 2

kTimestampTypeRsvd (C++ enumerator), 2

kTimestampTypeUnknown (C++ enumerator), 2

## L

LidarFeature (C++ type), 1

LidarGetExtrinsicParameter (C++ function), 23

LidarGetExtrinsicParameterCallback (C++ type), 23

LidarGetExtrinsicParameterResponse (C++ class), 22

LidarGetExtrinsicParameterResponse::pitch (C++ member), 22

LidarGetExtrinsicParameterResponse::ret\_code (C++ member), 22

LidarGetExtrinsicParameterResponse::roll (C++ member), 22

LidarGetExtrinsicParameterResponse::x (C++ member), 23

LidarGetExtrinsicParameterResponse::y (C++ member), 23

LidarGetExtrinsicParameterResponse::yaw (C++ member), 23

LidarGetExtrinsicParameterResponse::z (C++ member), 23

LidarMode (C++ type), 21

LidarModeRequestItem (C++ class), 12

LidarModeRequestItem::broadcast\_code (C++ member), 13

LidarModeRequestItem::state (C++ member), 13

LidarRainFogSuppress (C++ function), 23

LidarSetExtrinsicParameter (C++ function), 22

LidarSetExtrinsicParameterRequest (C++ class), 22

LidarSetExtrinsicParameterRequest::pitch (C++ member), 22

LidarSetExtrinsicParameterRequest::roll (C++ member), 22

LidarSetExtrinsicParameterRequest::x (C++ member), 22

LidarSetExtrinsicParameterRequest::y (C++ member), 22

LidarSetExtrinsicParameterRequest::yaw (C++ member), 22

LidarSetExtrinsicParameterRequest::z (C++ member), 22

LidarSetMode (C++ function), 21

LidarStartSampling (C++ function), 21

LidarState (C++ type), 1

LidarStateItem (C++ class), 13

LidarStateItem::broadcast\_code (C++ member), 13

LidarStateItem::error\_union (C++ member), 13

LidarStateItem::feature (C++ member), 13

LidarStateItem::state (C++ member), 13

LidarStopSampling (C++ function), 22

LivoxEthPacket (C++ class), 7

LivoxEthPacket::data (C++ member), 8

LivoxEthPacket::data\_type (C++ member), 8

LivoxEthPacket::err\_code (C++ member), 8

LivoxEthPacket::id (C++ member), 8

LivoxEthPacket::rsvd (C++ member), 8

LivoxEthPacket::slot (C++ member), 8

LivoxEthPacket::timestamp (C++ member), 8

LivoxEthPacket::timestamp\_type (C++ member), 8

LivoxEthPacket::version (C++ member), 8

## Q

QueryDeviceInformation (C++ function), 7

## R

RainFogSuppressRequestItem (C++ class), 18

RainFogSuppressRequestItem::broadcast\_code  
(C++ member), 18

RainFogSuppressRequestItem::feature  
(C++ member), 18

ResponseStatus (C++ type), 1

## S

SetBroadcastCallback (C++ function), 4

SetCartesianCoordinate (C++ function), 8

SetDataCallback (C++ function), 8

SetDeviceIPModeRequest (C++ class), 9

SetDeviceIPModeRequest::ip\_addr (C++  
member), 9

SetDeviceIPModeRequest::ip\_mode (C++  
member), 9

SetDeviceStateUpdateCallback (C++ func-  
tion), 4

SetErrorMessageCallback (C++ function), 9

SetSphericalCoordinate (C++ function), 9

SetStaticDynamicIP (C++ function), 9

Start (C++ function), 3

StatusUnion (C++ type), 3

StatusUnion::progress (C++ member), 3

StatusUnion::status\_code (C++ member), 3

## T

TimestampType (C++ type), 2

## U

Uninit (C++ function), 3