Formant SDK Reference

Formant

Mar 10, 2025

CONTENTS:

CHAPTER

ONE

AGENT SDK REFERENCE

This section outlines the usage for each method of the Formant Agent SDK.

class formant.sdk.agent.v1.**Client**(*agent_url='unix:///var/lib/formant/agent.sock'*, *enable_logging=True*, *ignore_throttled=False*, *ignore_unavailable=False*, *local_dev=False*, *thread_pool_size=10*)

A client for interacting with the Formant agent. Automatically handles connection and reconnection to the agent. There are methods for:

- Ingesting telemetry datapoints
- Creating events
- Handling commands
- Ingesting transform frames
- Reading application configuration
- Handling teleop control datapoints

Parameters

- **agent_url** – The address of the Formant agent API.
- **enable_logging** – If `True`, this client will log some information to `stdout`.
- **ignore_throttled** – If `True`, `telemetry datapoint throttle` errors will not raise Exceptions. Throttled datapoints are still valid for teleoperation.
- **ignore_unavailable** – If `True`, `Formant agent unavailable` errors will not raise Exceptions.

call_cloud(*endpoint*, *method*, *body*, *headers*, *require_formant_auth*, *buffer_call*, *is_retryable*, *retryable_status_codes=[]*)

Allows the user to call an endpoint of the Formant Admin API authenticated by the Formant agent instead of user credentials.

API calls which allow `device` authentication can buffer and retry calls. For more information, see the following documentation:

Use the Formant agent to authenticate API calls

Buffering and retrying API calls

Note: If buffering is enabled, you will not get a return value from this function.

Parameters

- **endpoint** (*str*) – Full URL of the endpoint to call (can be found at https://docs.formant.io/reference).
- **method** (*str*) – The HTTP method to use (e.g., "POST", "PUT", "GET", "PATCH", "DELETE").
- **headers** (*Dict[str, str]*) – Set the content type of your payload.
- **body** (*str*) – Payload of the request (parameters found at https://docs.formant.io/reference).
- **require_formant_auth** (*bool*) – Whether or not to use device authentication. If True, authorization header is added automatically.
- **buffer_call** (*bool*) – Whether or not to buffer the call. If True, the call is buffered and will be retried if necessary. If True, call_cloud() returns None.
- **is_retryable** (*bool*) – (buffer_call=True only) Whether to retry the call if it fails.
- **retryable_status_codes** (*List[int]*) – (buffer_call=True only) The status codes to retry on. A value of [-1] will retry on all 5xx codes EXCEPT FOR the following: [500, 501, 502, 505, 507, 508, 510, 511].

Return type

agent_pb2.PostGenericAPIUnbufferedRequestResponse

```
from formant.sdk.agent.v1 import Client
import json

fclient = Client()

payload = {
    "query": "acme",
    "count": 10
}

response = fclient.call_cloud(
    endpoint="https://api.formant.io/v1/admin/devices/query",
    method="POST",
    headers={
        "Content-Type": "application/json"
    },
    body=json.dumps(payload),
    require_formant_auth=True,
    buffer_call=False,
    is_retryable=False,
    retryable_status_codes=[]
)

# You get a response with ``statusCode`` and ``responseBody``
# when ``buffer_call == False``.
print(response.statusCode)
print(response.responseBody)
```

create_event(*message*, *tags=None*, *timestamp=None*, *end_timestamp=None*, *notify=False*, *severity='info'*)

Creates and ingests an event.

Parameters

- **message** (str) – The text payload of the event
- **tags** (Optional[Dict[str, str]]) – Tags to include on the event
- **timestamp** (Optional[int]) – Unix starting timestamp for the event. Uses the current time by default
- **end_timestamp** (Optional[int]) – Unix ending timestamp for the event. Must be greater than timestamp. If end_timestamp is supplied, the event will span a length of time
- **notify** (bool) – If True, the created event will trigger a Formant notification
- **severity** (('info', 'warning', 'critical', 'error')) – The severity level of the event

Return type
None

```
from formant.sdk.agent.v1 import Client

fclient = Client()
fclient.create_event(
    "Confinement beam to warp frequency 0.4e17 hz",
    tags={"Region": "North"},
    notify=True,
    severity="warning"
)
```

create_labeling_intervention_request(*title*, *instruction*, *labels*, *hint=None*, *url=None*, *content_type='image/jpg'*, *timestamp=None*, *severity='info'*)

Creates an intervention request based on type "labeling".

Parameters

- **title** – (str) The name of the intervention
- **instruction** – (str) The instructions detailing how to resolve the intervention
- **labels** – (Dict[str, str]) An Array of labels
- **hint** – (Optional[List[intervention_pb2.LabeledPolygon]]) An array of label polygons, X and Y coordinates with a label
- **url** – (str) The path to local file or valid remote URL for remote files
- **content_type** – (Literal["image/jpg", "image/png"]) The format of the encoded image or frame. Defaults to "image/jpg".
- **timestamp** – (Optional[int]) Unix timestamp in milliseconds for the posted datapoint. Uses the current time by default
- **severity** – (Literal["info", "warning", "critical", "error"]) The severity level of the event

Return type
intervention_pb2.InterventionRequest

Each label in labels defined as:

```
Label = {
        value = string;
```

(continues on next page)

(continued from previous page)

```
        string display_name = string;
        }
```

Hint is an array of "LabeledPolygon", defined as:

```
hint = {
    List of vertex,
    List of labels
}
```

where each vertex is defined as:

```
vertex = {
    x = float,
    y = float
}
```

create_selection_intervention_request(*title*, *instruction*, *options*, *hint*, *url=None*, *content_type='image/jpg'*, *timestamp=None*, *severity='info'*)

Creates an intervention request based on type `selection`. Takes an image url, options and an integer with an optional addition of instructions, and title.

Parameters

- **title** – (str) The name of the intervention
- **instruction** – (str) The instructions detailing how to resolve the intervention
- **options** – (List[str]) The list with options to select from
- **hint** – (int) The index of the suspected correct answer
- **url** – (str) The path to local file or valid remote URL for remote files
- **content_type** – (Literal["image/jpg", "image/png"]) The format of the encoded image or frame. Defaults to `"image/jpg"`
- **timestamp** – (Optional[int]) Unix timestamp in milliseconds for the posted datapoint. Uses the current time by default
- **severity** – (Literal["info", "warning", "critical", "error"]) The severity level of the event

Return type

InterventionRequest

```
from formant.sdk.agent.v1 import Client

fclient = Client()
fclient.create_selection_intervention_request(
    "Which fruit is best?",
    "Select the best grape",
    ["fruit_1", "fruit_2", "fruit_3"],
    hint=1,
    url=/home/my_user/data/test-image.jpeg
    severity=critical
)
```

custom_data_channel_binary_request_handler(*channel_name*, *new_thread=False*)

Parameters
channel_name – The name of the custom data channel to listen on.

```
from formant.sdk.agent.v1 import Client

fclient = Client()

@fclient.custom_data_channel_request_handler("my_channel")
def handler(request_data):
    # Do something with request_data bytes
    print(request_data.decode("utf-8"))

    # Return any bytes response
    return b"Hello."
```

custom_data_channel_request_handler(*channel_name*)

Registers a handler for requests sent by RequestDataChannel instances (part of the Formant toolkit). See: https://github.com/FormantIO/toolkit/tree/master/examples/request-response for an example.

Parameters
channel_name – The name of the custom data channel to listen on.

Return type
GetCustomDataChannelMessageStreamResponse

```
from formant.sdk.agent.v1 import Client

fclient = Client()

@fclient.custom_data_channel_request_handler("my_channel")
def handler(request_data):
    # Do something with request_data string
    print(json.loads(request_data))

    # Return any string response
    return json.dumps({"message": "Hello world!"})
```

get_agent_id()

Gets the Device ID for this device.

Return type
str

get_app_config(*key*, **args*)

Returns the value for the given key that was set in Formant application configuration for this device, or returns the given default value.

Parameters

- **key** (str) – The application configuration key
- **args** (Any) – (One additional argument) The default value to return if the key is not found.

Raises
TypeError: Function takes at most two args: (key: str, default: Any)

Return type
Optional[str]

`get_buffer_metadata`()

Returns the current WebRTC buffer statistics.

Return type
agent_pb2.GetBufferMetadata

`get_command_request`(*command_filter=None*)

If there is a command request in the agent's queue whose `command` value matches an element of the given command filter, takes and returns the command request. Otherwise, returns `None` if there are no matching command requests in the agent's queue.

Parameters
command_filter (Optional[List[str]]) – A list of command names. This method only returns commands whose names are in this list.

Return type
CommandRequest, None

`get_config_blob_data`()

Returns the blob data defined in the device configuration.

Return type
str

`get_intervention_response`(*request_id*, *timeout=None*)

Receives request ID, and returns a response.

Parameters

- **request_id** (*str*) – The ID of the intervention request to which this method responds
- **timeout** (*int*) – (Optional) Number of seconds to wait for a response.

Return type
agent_pb2.GetInterventionResponse

```
from formant.sdk.agent.v1 import Client

fclient = Client()
request = fclient.create_selection_intervention_request(
    title="",
    instruction="instruction",
    options=["option1", "option2", "option3"],
    hint=0,
    url="/home/formantuser/Downloads/image.png",
)
# Waits 5 seconds for a response, then proceeds
response = fclient.get_intervention_response(request.id, 5)
```

`get_teleop_info`()

Returns current information about teleop connection count.

Return type
GetTeleopInfoResponse

post_battery(*stream*, *percentage*, *voltage=None*, *current=None*, *charge=None*, *tags=None*, *timestamp=None*)

Post a battery datapoint to a telemetry stream. Only percentage is required.

Parameters

- **stream** (str) – The name of the Formant stream to post the datapoint on
- **percentage** (Union[int, float]) – The battery charge percentage
- **voltage** (Union[int, float, None]) – The battery voltage
- **current** (Union[int, float, None]) – The battery current
- **charge** (Union[int, float, None]) – The battery charge
- **tags** (Optional[Dict[str, str]]) – Tags to include on the posted datapoint
- **timestamp** (Optional[int]) – Unix timestamp in milliseconds for the posted datapoint. Uses the current time by default

Return type
None

post_bitset(*stream*, *bitset_dict*, *tags=None*, *timestamp=None*)

Post a bitset datapoint to a telemetry stream. A bitset is a collection of related boolean states.

Parameters

- **stream** (str) – The name of the Formant stream to post the datapoint on
- **bitset_dict** (Dict[str, bool]) – The datapoint value, a dictionary mapping names to booleans
- **tags** (Optional[Dict[str, str]]) – Tags to include on the posted datapoint
- **timestamp** (Optional[int]) – Unix timestamp in milliseconds for the posted datapoint. Uses the current time by default

Return type
None

```
from formant.sdk.agent.v1 import Client

fclient = Client()
fclient.post_bitset(
    "example.bitset",
    {
        "standing": False,
        "walking": False,
        "sitting": True
    }
)
```

post_file(*stream*, *url=None*, *filename=None*, *tags=None*, *timestamp=None*)

Post a file to a telemetry stream.

Parameters

- **stream** (str) – The name of the Formant stream to post the file on
- **url** (Optional[str]) – The file url: path to local file or valid remote URL for remote files

- **filename** (Optional[str]) – The file name: name displayed inside Formant module
- **tags** (Optional[Dict[str, str]]) – Tags to include on the posted file
- **timestamp** (Optional[int]) – Unix timestamp in milliseconds for the posted file. Uses the current time by default

Return type
None

```
from formant.sdk.agent.v1 import Client

fclient = Client()
fclient.post_file(
    "example.file",
    /home/user/Desktop/data/planets.csv,
    planets.csv,
)
```

post_geolocation(*stream*, *latitude*, *longitude*, *tags=None*, *timestamp=None*, *altitude=None*, *orientation=None*)

Post a geolocation datapoint to a telemetry stream.

Parameters

- **stream** (str) – The name of the Formant stream to post the datapoint on
- **latitude** (Union[float, int]) – The datapoint value's latitude
- **longitude** (Union[float, int]) – The datapoint value's longitude
- **tags** (Optional[Dict[str, str]]) – Tags to include on the posted datapoint
- **timestamp** (Optional[int]) – Unix timestamp in milliseconds for the posted datapoint. Uses the current time by default

Return type
None

post_image(*stream*, *value=None*, *url=None*, *content_type='image/jpg'*, *tags=None*, *timestamp=None*)

Post an image datapoint to a telemetry stream.

Parameters

- **stream** (str) – The name of the Formant stream to post the datapoint on
- **value** (Optional[bytes]) – The datapoint value: raw bytes of an encoded image or frame
- **url** (Optional[str]) – The datapoint url: path to local file or valid remote URL for remote files
- **content_type** (('image/jpg', 'image/png', 'video/h264')) – The format of the encoded image or frame. Defaults to image/jpg.
- **tags** (Optional[Dict[str, str]]) – Tags to include on the posted datapoint
- **timestamp** (Optional[int]) – Unix timestamp in milliseconds for the posted datapoint. Uses the current time by default

Return type
None

post_json(*stream*, *value*, *tags=None*, *timestamp=None*)

Post a JSON datapoint to a telemetry stream.

Parameters

- **stream** (str) – The name of the Formant stream to post the datapoint on
- **value** (str) – The encoded JSON datapoint value
- **tags** (Optional[Dict[str, str]]) – Tags to include on the posted datapoint
- **timestamp** (Optional[int]) – Unix timestamp in milliseconds for the posted datapoint. Uses the current time by default

Return type
None

post_numeric(*stream*, *value*, *tags=None*, *timestamp=None*)

Post a numeric datapoint to a telemetry stream.

Parameters

- **stream** (str) – The name of the Formant stream to post the datapoint on
- **value** (Union[float, int]) – The numeric datapoint value
- **tags** (Optional[Dict[str, str]]) – Tags to include on the posted datapoint
- **timestamp** (Optional[int]) – Unix timestamp in milliseconds for the posted datapoint. Uses the current time by default

Return type
None

post_numericset(*stream*, *numerics_dict*, *tags=None*, *timestamp=None*)

Post a numeric set datapoint to a telemetry stream. Numeric sets are collections of related numeric datapoints.

Parameters

- **stream** (str) – The name of the Formant stream to post the datapoint on
- **numerics_dict** (Dict[str, Tuple[Union[float, int], Optional[str]]]) – The numeric set datapoint value, a dictionary mapping names to (numeric value, units) tuples.
- **tags** (Optional[Dict[str, str]]) – Tags to include on the posted datapoint
- **timestamp** (Optional[int]) – Unix timestamp in milliseconds for the posted datapoint. Uses the current time by default

Return type
None

```
from formant.sdk.agent.v1 import Client

fclient = Client()
fclient.post_numericset(
    "example.numericset2",
    {
        "frequency": (998, "Hz"),
        "usage": (30, "percent"),
        "warp factor": (6.0, None),
```

(continues on next page)

(continued from previous page)

```
    },
)
```

post_task_summary(*task_summary_format_id*, *task_summary_report*, *message*, *task_id*, *start_time*, *end_time=None*, *task_summary_url='https://api.formant.io/v1/admin/task-summaries/'*, *additional_request_kwargs={}*)

Uploads a task summary to the Formant cloud in the provided task summary format.

You must first create a task summary format and add it to Formant. See Create a task summary.

Parameters

- **task_summary_format_id** (*str*) – ID of the task summary format which describes this task summary.
- **task_summary_report** (*dict*) – Data for this task summary in key-value pairs, as described by the task summary format.
- **message** (*str*) – Message associated with this task summary.
- **task_id** (*str*) – Enter a unique identifier for this task summary.
- **start_time** (*str*) – Start datetime of the data range relevant to this event (ISO 8601 format).
- **end_time** (*str*) – (Optional) End time of the data range relevant to this event.
- **task_summary_url** (*str*) – (Optional) The URL to which to post the task summary.
- **additional_request_kwargs** (*dict*) – (Optional) Additional request kwargs to pass in the POST request. These will be added to the body of the request as key-value pairs. See all available request kwargs at Task Summary POST.

Return type

Dictionary containing task summary

post_text(*stream*, *value*, *tags=None*, *timestamp=None*)

Post a text datapoint to a stream.

Parameters

- **stream** (str) – The name of the Formant stream to post the datapoint on
- **value** (str) – The text datapoint value
- **tags** (Optional[Dict[str, str]]) – Tags to include on the posted datapoint
- **timestamp** (Optional[int]) – Unix timestamp in milliseconds for the posted datapoint. Uses the current time by default

Return type

None

```
from formant.sdk.agent.v1 import Client

fclient = Client()
fclient.post_text(
    "example.text",
    "Processed 9 items"
)
```

post_transform_frame(*parent_frame*, *child_frame*, *tx*, *ty*, *tz*, *rx*, *ry*, *rz*, *rw*)

Adds a transform frame, used to position datapoints in 3D space.

Parameters

- **parent_frame** (str) – The parent frame of the posted transform
- **child_frame** (str) – The child frame of the posted transform
- **tx** (Union[int, float]) – x-translation
- **ty** (Union[int, float]) – y-translation
- **tz** (Union[int, float]) – z-translation
- **rx** (Union[int, float]) – x-rotation (quaternion)
- **ry** (Union[int, float]) – y-rotation (quaternion)
- **rz** (Union[int, float]) – z-rotation (quaternion)
- **rw** (Union[int, float]) – w-rotation (quaternion)

Return type
None

prepare_battery(*stream*, *percentage*, *voltage=None*, *current=None*, *charge=None*, *tags=None*, *timestamp=None*)

Prepare a battery datapoint without posting it.

Parameters

- **stream** – (str) The name of the Formant stream for the datapoint
- **percentage** – (Union[int, float]) The battery charge percentage
- **voltage** – (Optional[Union[int, float]]) The battery voltage (optional)
- **current** – (Optional[Union[int, float]]) The battery current (optional)
- **charge** – (Optional[Union[int, float]]) The battery charge (optional)
- **tags** – (Optional[Dict[str, str]]) Tags for the datapoint
- **timestamp** – (Optional[int]) Unix timestamp in milliseconds for the datapoint. Uses the current time by default

Returns
The prepared battery datapoint

Return type
datapoint_pb2.Datapoint

prepare_bitset(*stream*, *bitset_dict*, *tags=None*, *timestamp=None*)

Prepare a bitset datapoint without posting it.

Parameters

- **stream** – (str) The name of the Formant stream for the datapoint
- **bitset_dict** – (Dict[str, bool]) The datapoint value, a dictionary mapping names to booleans
- **tags** – (Optional[Dict[str, str]]) Tags for the datapoint
- **timestamp** – (Optional[int]) Unix timestamp in milliseconds for the datapoint. Uses the current time by default

Return type
datapoint_pb2.Datapoint

`prepare_file`(*stream*, *url=None*, *filename=None*, *tags=None*, *timestamp=None*)

Prepare a file datapoint without posting it.

Parameters

- **stream** – (str) The name of the Formant stream for the datapoint
- **url** – (str) The file url: path to a local file or valid remote URL for remote files
- **filename** – (Optional[str]) The file name: name displayed inside Formant module
- **tags** – (Optional[Dict[str, str]]) Tags for the datapoint
- **timestamp** – (Optional[int]) Unix timestamp in milliseconds for the datapoint. Uses the current time by default

Returns
The prepared file datapoint

Return type
datapoint_pb2.Datapoint

`prepare_geolocation`(*stream*, *latitude*, *longitude*, *tags=None*, *timestamp=None*, *altitude=None*, *orientation=None*)

Prepare a geolocation datapoint without posting it.

Parameters

- **stream** – (str) The name of the Formant stream for the datapoint
- **latitude** – (Union[float, int]) The datapoint value's latitude
- **longitude** – (Union[float, int]) The datapoint value's longitude
- **tags** – (Optional[Dict[str, str]]) Tags for the datapoint
- **timestamp** – (Optional[int]) Unix timestamp in milliseconds for the datapoint. Uses the current time by default
- **altitude** – (Union[float, int]) The altitude value (optional)
- **orientation** – (Union[float, int]) The orientation value (optional)

Returns
The prepared geolocation datapoint

Return type
datapoint_pb2.Datapoint

`prepare_image`(*stream*, *value=None*, *url=None*, *content_type='image/jpg'*, *tags=None*, *timestamp=None*)

Prepare an image datapoint without posting it.

Parameters

- **stream** – (str) The name of the Formant stream for the datapoint
- **value** – (Optional[bytes]) The datapoint value: raw bytes of an encoded image or frame
- **url** – (Optional[str]) The datapoint url: path to a local file or valid remote URL for remote files
- **content_type** – (Literal["image/jpg", "image/png", "video/h264"]) The format of the encoded image or frame. Defaults to `"image/jpg"`.

- **tags** – (Optional[Dict[str, str]]) Tags for the datapoint
- **timestamp** – (Optional[int]) Unix timestamp in milliseconds for the datapoint. Uses the current time by default

Return type
datapoint_pb2.Datapoint

Raises
`InvalidArgument: One of [url, value] must be used.`

prepare_json(*stream*, *value*, *tags=None*, *timestamp=None*)

Prepare a JSON datapoint without posting it.

Parameters

- **stream** – (str) The name of the Formant stream for the datapoint
- **value** – (str) The encoded JSON datapoint value
- **tags** – (Optional[Dict[str, str]]) Tags for the datapoint
- **timestamp** – (Optional[int]) Unix timestamp in milliseconds for the datapoint. Uses the current time by default

Return type
datapoint_pb2.Datapoint

prepare_numeric(*stream*, *value*, *tags=None*, *timestamp=None*)

Prepare a numeric datapoint without posting it.

Parameters

- **stream** – (str) The name of the Formant stream for the datapoint
- **value** – (Union[float, int]) The numeric datapoint value
- **tags** – (Optional[Dict[str, str]]) Tags for the datapoint
- **timestamp** – (Optional[int]) Unix timestamp in milliseconds for the datapoint. Uses the current time by default

Return type
datapoint_pb2.Datapoint

prepare_numericset(*stream*, *numerics_dict*, *tags=None*, *timestamp=None*)

Prepare a numeric set datapoint without posting it.

Parameters

- **stream** – (str) The name of the Formant stream for the datapoint
- **numerics_dict** – (Dict[str, Tuple[Union[float, int], Optional[str]]]) The numeric set datapoint value, a dictionary mapping names to (numeric value, units) tuples.
- **tags** – (Optional[Dict[str, str]]) Tags for the datapoint
- **timestamp** – (Optional[int]) Unix timestamp in milliseconds for the datapoint. Uses the current time by default

Return type
The prepared numeric set datapoint

Raises
`TypeError: value v for key k in numericset must have length of 2`

prepare_text(*stream*, *value*, *tags=None*, *timestamp=None*)

Prepare a text datapoint without posting it.

Parameters

- **stream** (str) – The name of the Formant stream for the datapoint
- **value** (str) – The text datapoint value
- **tags** (Optional[Dict[str, str]]) – Tags for the datapoint
- **timestamp** (Optional[int]) – Unix timestamp in milliseconds for the datapoint. Uses the current time by default

Return type
datapoint_pb2.Datapoint

register_command_request_callback(*f*, *command_filter=None*)

Command requests issued to the agent whose command value matches an element of the given command filter will be streamed into the provided callback. If no command filter is provided, all command requests will be handled.

Parameters

- **f** (Callable[[CommandRequest], None]) – A callback that will be executed on command requests as they are received by the Formant agent.
- **command_filter** (Optional[List[str]]) – A list of command names. The provided callback is only executed on commands whose names are in this list

Return type
None

register_config_update_callback(*f*)

Adds a function to the list of callbacks that are executed by the client when this device receives updated configuration from Formant.

Parameters
f (Callable) – The configuration update callback to be registered.

Return type
None

register_custom_data_channel_message_callback(*f*, *channel_name_filter=None*)

Registers a callback on data presence on the specified data channel.

Parameters

- **f** – A callback that will be called with messages received on the specified custom data channel.
- **channel_name_filter** – An optional allow list of custom channel names for this callback.

Return type
None

register_telemetry_listener_callback(*f*, *stream_filter=None*)

Datapoints posted to the Formant agent whose "stream" value matches an element of the given stream filter will be streamed into the provided callback. If no stream filter is provided, datapoints from all streams will be received.

Parameters

- **f** – A callback that will be called when a datapoint is posted to the Formant agent
- **stream_filter** – A list of stream names. The provided callback is only called for datapoints whose stream name is in this list

register_teleop_callback(*f*, *stream_filter=None*)

Control datapoints received from teleop whose `stream` value matches an element of the given stream filter will be streamed into the provided callback. If no stream filter is provided, control datapoints from all streams will be received.

Parameters

- **f** (`Callable[[ControlDatapoint], None]`) – A callback that will be executed on teleop control datapoints as they are received by the Formant agent
- **stream_filter** (`Optional[List[str]]`) – A list of stream names. The provided callback is only exectued on control datapoints whose names are in this list

Return type
None

register_teleop_heartbeat_callback(*f*)

The provided callback will be called once each time a heartbeat is received over Formant teleop. Heartbeats are streamed from the operator machine at 20Hz on a UDP-like channel. This method can be used to quickly detect teleop disconnections.

Parameters
f – A callback that will be called when a heartbeat is received.

Return type
None

send_command_response(*request_id*, *success*, *datapoint=None*)

Sends a command response for an identified command request to Formant. Returns an error if there was a problem sending the command response.

Parameters

- **request_id** (`str`) – The ID of the command request to which this method responds
- **success** (`bool`) – Whether the command was successfully executed
- **datapoint** (`Optional[Datapoint]`) – A datapoint related to the command. Can attach a datapoint to a command response. E.g., if a command fails, can ingest a text datapoint with an error message related to the failure of the command.

Return type
None

send_on_custom_data_channel(*channel_name*, *payload*)

Sends data on custom data channel.

Parameters

- **channel_name** – (str) The name of the channel over which to send data
- **payload** – (bytes) The data payload to send.

Return type
None

`set_base_frame_id`(*base_reference_frame*)

Sets the base reference frame for tf tree ingestion.

Parameters
`base_reference_frame` – The base reference frame for the tf tree.

Return type
None

`unregister_command_request_callback`(*f*)

Unregisters previously registered command request callback.

Parameters
`f` (`Callable`[[`CommandRequest`], `None`]) – The command request callback to be unregistered

Return type
None

`unregister_config_update_callback`(*f*)

Removes a function from the list of callbacks that are executed by the client when this device receives updated configuration from Formant.

Parameters
`f` (`Callable`) – The configuration update callback to be unregistered.

Return type
None

`unregister_custom_data_channel_message_callback`(*f*)

Unregisters previously registered custom data channel callback.

Parameters
`f` – The custom data channel message callback to be unregistered.

Return type
None

`unregister_telemetry_listener_callback`(*f*)

Unregisters previously registered telemetry loopback callback.

Parameters
`f` – The telemetry loopback callback to be unregistered

`unregister_teleop_callback`(*f*)

Unregisters previously registered teleop callback.

Parameters
`f` (`Callable`[[`ControlDatapoint`], `None`]) – The teleop callback to be unregistered

Return type
None

`unregister_teleop_heartbeat_callback`(*f*)

Unregisters previously registered teleop heartbeat callback.

Parameters
`f` – The teleop heartbeat callback to be unregistered

Return type
None

CHAPTER

TWO

CLOUD SDK V1 REFERENCE

This section outlines the usage for each method of the Formant Cloud SDK v1.

class formant.sdk.cloud.v1.**Client**(*admin_api='https://api.formant.io/v1/admin'*, *ingest_api='https://api.formant.io/v1/ingest'*, *query_api='https://api.formant.io/v1/queries'*)

A client for interacting with the Formant Cloud. There are methods for:

- Ingesting telemetry datapoints for device(s)
- Query telemetry datapoints
- Query stream(s) last known value
- Create intervention requests
- Create intervention responses

To authenticate the Cloud SDK v1 client, set the following environment variables with valid Formant credentials:

- `FORMANT_EMAIL`
- `FORMANT_PASSWORD`

Return values

All methods of the Cloud SDK v1 client return a Dictionary object which can be parsed for response values.

---

create_adapter(*params*)

Create an adapter in your organization. Full parameters can be found here: Adapter POST.

```
from formant.sdk.cloud.v1 import Client

fclient = Client()

params = {
    "execCommand": "./start.sh",
    "path": "/tmp/model.dat"
    "name": "adapters_name"
}

response = fclient.create_adapter(params)
```

create_command(*params*)

Create a command. Full parameters can be found here: Command template POST.

```
from formant.sdk.cloud.v1 import Client

fclient = Client()

params = {
    deviceId: "abc-123"
    command: "return_to_charge_station"
    parameter: {
        "scrubberTime": "2014-11-03T19:38:34.203Z",
        "value": "A-2",
        "files": [{
            "id": "def-456",
            "name": "optional_name1"
        }]
    },
}

response = fclient.create_command(params)
```

create_device(*device_name*, *publicKey=''*, *tags=None*, *params=None*)

Creates a new device.

Parameters
device_name (str) – Device name.

create_intervention_request(*params*)

Create an intervention request. Full parameters can be found here: Intervention request POST.

Parameters
params – Intervention request parameters.

```
from formant.sdk.cloud.v1 import Client

fclient = Client()

params = {
    "message": "A teleop for a customer is requested",
    "interventionType": "teleop",
    "time": "2022-02-17T11:41:33.389-08:00",
    "deviceId": "b306de84-33ca-4917-9218-f686730e24e0",
    "tags": {},
    "data": {
        "instruction": "Look at the users item on the table"
    }
}

response = fclient.create_intervention_request(params)
```

create_intervention_response(*params*)

Creates a response to an intervention request. Full parameters can be found here: Intervention response POST.

Parameters
params – Intervention response parameters.

```
from formant.sdk.cloud.v1 import Client

fclient = Client()

params = {
"interventionId": "518e24fc-64ef-47bb-be5e-036a97aeafaa",
"interventionType": "teleop",
"data": {
    "state": "success",
    "notes": "looks good!"
    }
}

response = fclient.create_intervention_response(params)
```

generate_provisioning_token(*id*, *params=None*)

Generates a provisioning token for a device.

Parameters
id (str) – ID of the device to provision.

get_annotation_templates()

Gets all annotation templates in this organization.

get_organization()

Get this organization ID.

get_organization_id()

Gets this organization ID.

Returns
Organization ID.

Return type
str

get_user_id()

Gets self user ID.

Returns
ID of this user.

Return type
str

ingest(*params*)

Ingests data to Formant.

Note: Administrator credentials required.

```
from formant.sdk.cloud.v1 import Client

fclient = Client()

params = {
```

(continues on next page)

(continued from previous page)

```
    deviceId: "ced176ab-f223-4466-b958-ff8d35261529",
    name: "engine_temp",
    type: "numeric",
    tags: {"location":"sf"},
    points: [...],
}

response = fclient.ingest(params)
```

patch_device(*device_id*, *params*)

Update device configuration. Full parameters can be found here: Device PATCH.

Parameters

- **device_id** (str) – ID of the device to update.
- **params** – Device configuration parameters to update.

```
from formant.sdk.cloud.v1 import Client

fclient = Client()

device_id = 'abc-123'
params = {
    "desiredConfiguration": 43
}

response = fclient.patch_device(device_id, params)
```

post_device_configuration(*device_id*, *params*)

Post a device configuration.

```
from formant.sdk.cloud.v1 import Client

fclient = Client()

device_id = 'abc-123'
params = {
    "document": {
        adapter: [{
            id: "84f98678-5f18-478d-aed8-631d9ea043a9",
            name: "ROS-diagnostics",
            "execCommand": "./start.sh"
            }],
        tags: {},
        telemetry: {
            streams: []
        }
    }

response = fclient.post_device_configuration(device_id, params)
```

provision_device(*provisioningToken*, *publicKey*, *params=None*)

Provision a device given an ID and a provisioning token.

Parameters
provisioningToken (str) – Provisioning token from generate_provisioning_token.

query(*params*)

Queries datapoints from the Formant cloud. For more information, see Cloud SDK: Querying telemetry data.

```
from formant.sdk.cloud.v1 import Client

fclient = Client()

params = {
    start: "2021-01-01T01:00:00.000Z",
    end: "2021-01-01T02:00:00.000Z",
    deviceIds: ["99e8ee37-0a27-4a11-bba2-521facabefa3"],
    names: ["engine_temp"],
    types: ["numeric"],
    tags: {"location":["sf","la"]},
    notNames: ["speed"],
}

response = fclient.query(params)
```

query_commands(*params*)

Get undelivered commands by device ID.

```
from formant.sdk.cloud.v1 import Client

fclient = Client()

params = {
deviceId: "abc-123",
}

response = fclient.query_commands(params)
```

query_devices(*params*)

Query devices in this organization. The full list of query parameters can be found here: Device QUERY.

Parameters
params (*object*) – Query parameters.

```
from formant.sdk.cloud.v1 import Client

fclient = Client()

params = {
    name: "model00.001",
    tags: {"location":["sf", "la"]},
}

response = fclient.query_devices(params)
```

query_stream_current_value(*params*)

Get current value for streams which match query parameters. Full parameters can be found here: Stream

current value QUERY

```
from formant.sdk.cloud.v1 import Client

fclient = Client()

params = {
    start: "2021-01-01T01:00:00.000Z",
    end: "2021-01-01T02:00:00.000Z",
    deviceIds: ["99e8ee37-0a27-4a11-bba2-521facabefa3"],
    names: ["engine_temp"],
    types: ["numeric"],
    tags: {"location":["sf","la"]},
    notNames: ["speed"],
}

response = fclient.query_stream_current_value(params)
```

query_task_summaries(*params*)

Get all task summaries

query_task_summary_formats()

Get all task summary formants

upload_file(*params*)

Upload a file to the Formant cloud.

```
from formant.sdk.cloud.v1 import Client

fclient = Client()

params = {
    path: "/tmp/model.dat"
}

response = fclient.upload_file(params)
```

upload_task_summary(*task_summary_format_id*, *report*, *device_id*, *tags=None*, *message=None*)

Upload a task summary.

Task summary definition can be found here: Task summary POST.

upload_task_summary_format(*task_summary_format*)

Upload a task summary format.

Task summary format definition can be found here: Task summary format POST.

INDEX

Q

R

S

U