# Formant SDK Reference

Formant

Dec 03, 2024

# 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.

**get_agent_id**()

Gets the Device ID for this device.

**Return type**
str

**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_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),
    },
)
```

**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_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",
```

(continues on next page)

(continued from previous page)

```
    {
        "standing": False,
        "walking": False,
        "sitting": True
    }
)
```

**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_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[float, int, None]) – The battery voltage
- **current** (Union[float, int, None]) – The battery current
- **charge** (Union[float, int, 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_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,
)
```

**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

**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_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_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_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_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_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

**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

**unregister_telemetry_listener_callback**(*f*)

Unregisters previously registered telemetry loopback callback.

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

**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

**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

**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"
)
```

**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

**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

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

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

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

unregister_teleop_callback(*f*)

Unregisters previously registered teleop callback.

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

**Return type**
None

get_teleop_info()

Returns current information about teleop connection count.

**Return type**
GetTeleopInfoResponse

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

`unregister_teleop_heartbeat_callback`(*f*)

Unregisters previously registered teleop heartbeat callback.

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

**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

`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

`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

`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

```python
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!"})
```

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

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

```python
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."
```

**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_config_blob_data**()

Returns the blob data defined in the device configuration.

**Return type**
str

**get_buffer_metadata**()

Returns the current WebRTC buffer statistics.

**Return type**
agent_pb2.GetBufferMetadata

`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

`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

`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

```python
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_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
)
```

**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;
        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,
```

(continues on next page)

(continued from previous page)

```
    y = float
}
```

**get_intervention_response**(*request_id*)

Receives request ID, and returns a response.

**Parameters**

- **request_id** – (str) The ID of the intervention request to which this method responds
- **timestamp** – Unix timestamp in milliseconds for the posted datapoint. Uses the current time by default.

**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",
)
fclient.handle_intervention_response(request.id)
```

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

Requires service account credentials (environment variables):

- FORMANT_EMAIL
- FORMANT_PASSWORD

`FORMANT_EMAIL` and `FORMANT_PASSWORD` environment variables must be set with valid credentials.

**Raises**

- **ValueError** – `Missing FORMANT_EMAIL environment variable`
- **ValueError** – `Missing FORMANT_PASSWORD environment variable`

**get_user_id**()

Gets self user ID.

**Returns**
ID of this user.

**Return type**
`str`

**get_organization_id**()

Gets this organization ID.

**Returns**
Organization ID.

**Return type**
`str`

**ingest**(*params*)

Ingests data to Formant.

**Note:** Administrator credentials required.

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

fclient = Client()

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

response = fclient.ingest(params)
```

**get_organization**()

Get this organization ID.

**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_devices**(*params*)

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

**Parameters**
**params** (*object*) – Query parameters.

**Returns**
Query response.

**Return type**
object

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

fclient = Client()

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

response = fclient.query_devices(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** (*obj*) – Device configuration parameters to update.

**Returns**

_description_

**Return type**

_type_

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

fclient = Client()

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

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

**query_task_summary_formats**()

Get all task summary formants

**query_task_summaries**(*params*)

Get all task summaries

**upload_task_summary_format**(*task_summary_format*)

Upload a task summary format.

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

**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.

**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)
```

**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)
```

**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)
```

**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)
```

**create_intervention_response**(*params*)

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

**Parameters**
**params** (*obj*) – Intervention response parameters.

**Returns**
_description_

**Return type**
_type_

```
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)
```

**create_intervention_request**(*params*)

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

**Parameters**
**params** (*obj*) – Intervention request parameters.

**Returns**
_description_

**Return type**
_type_

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

fclient = Client()

params = {
    "message": "A teleop for a customer is requested",
```

(continues on next page)

(continued from previous page)

```
    "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_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)
```

**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)
```

**get_annotation_templates**()

Gets all annotation templates in this organization.

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

Creates a new device.

**Parameters**
**device_name** (str) – Device name.

**Returns**
_description_

**Return type**
_type_

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

Generates a provisioning token for a device.

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

**Returns**
Provisioning token.

**Return type**
str

**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.

**Returns**
_description_

**Return type**
_type_

CHAPTER

THREE

# CLOUD SDK V2 REFERENCE

This section outlines the usage of the Cloud SDK v2.

The first section describes the Formant Cloud SDK v2 Client, which will be used to instantiate and authenticate a session with the Formant cloud.

Then, each object which can be manipulated in the Formant cloud is listed.

**class** formant.sdk.cloud.v2.client.**Client**(*email=None*, *password=None*, *base_url='https://api.formant.io/v1'*)

Creates a client to interact with the Formant cloud. Allows you to create, update, and delete entities in your Formant organization.

```
from formant.sdk.cloud.v2 import Client
fclient = Client()
```

**Parameters**

- **email** (*str, optional*) – Formant email address. This must be provided, or the FORMANT_EMAIL environment variable must be set. Defaults to None
- **password** (*str, optional*) – Formant password. This must be provided, or the FORMANT_PASSWORD environment variable must be set. Defaults to None
- **base_url** (*str, optional*) – API base URL, defaults to DEFAULT_BASE_URL

**Raises**

- **ValueError** – email argument missing and FORMANT_EMAIL environment variable not set!
- **ValueError** – password argument missing and FORMANT_PASSWORD environment variable not set

**class** formant.sdk.cloud.v2.src.resources.adapters.adapters.**Adapters**(*get_client*)

**create**(*adapter*)

Creates an Adapter

**async create_async**(*adapter*)

Creates an Adapter

**delete**(*id*)

Deletes and Adapter

async delete_async(*id*)

Deletes and Adapter

class formant.sdk.cloud.v2.src.resources.annotations.annotations.Annotations(*get_client*)

get_template(*id*)

Get an annotation

async get_template_async(*id*)

Get an annotation

list_templates()

List all annotations

async list_templates_async()

List all annotations

post(*annotation*)

Creates an annotation

async post_async(*annotation*)

Creates an annotation

class formant.sdk.cloud.v2.src.resources.commands.commands.Commands(*get_client*)

create(*command*)

Creates a command

async create_async(*command*)

Creates a command

query(*command_query*)

Query undelivered commands by device ID

async query_async(*command_query*)

Query undelivered commands by device ID

class formant.sdk.cloud.v2.src.resources.count.count.Count(*get_client*)

active_devices(*active_devices_query*)

Gets all the active devices during the timestamp

async active_devices_async(*active_devices_query*)

Gets all the active devices during the timestamp

class formant.sdk.cloud.v2.src.resources.devices.devices.Devices(*get_client*)

get_device(*device_id*)

Get a device

async get_device_async(*device_id*)

Get a device

get_device_configuration(*device_id*, *desired_configuration_version*)

Get a device configuration

async get_device_configuration_async(*device_id*, *desired_configuration_version*)

Get a device configuration

**patch**(*device_id*, *partial_device*)

Update a device

**async patch_async**(*device_id*, *partial_device*)

Update a device

**post_device_configuration**(*device_id*, *device_configuration*)

Create a device configuration

**async post_device_configuration_async**(*device_id*, *device_configuration*)

Create a device configuration

**query**(*device_query*)

Query devices by name and/or tags

**async query_async**(*device_query*)

Query devices by name and/or tags

**class** formant.sdk.cloud.v2.src.resources.events.events.**Events**(*get_client*)

**query**(*event_query*)

Get an event

**async query_async**(*event_query*)

Get an event

**class** formant.sdk.cloud.v2.src.resources.files.files.**Files**(*get_client*)

**upload**(*path*, *timeout=60*)

Uploads a file

**async upload_async**(*path*, *timeout=60*)

Uploads a file

**class** formant.sdk.cloud.v2.src.resources.ingest.ingest.**Ingest**(*get_client*)

**post**(*ingestion_request*)

**post_all**(*ingestion_requests*)

**async post_all_async**(*ingestion_requests*)

**async post_async**(*ingestion_request*)

**class** formant.sdk.cloud.v2.src.resources.metadata.metadata.**Metadata**(*get_client*)

**list_device_ids**(*scope_filer*)

List device ids

**async list_device_ids_async**(*scope_filer*)

List device ids

**list_metadata**(*scope_filer*)

List stream metadata

**async list_metadata_async**(*scope_filer*)

List stream metadata

**list_stream_names**(*scope_filer*)

List stream names

**async list_stream_names_async**(*scope_filer*)

List stream names

**class** formant.sdk.cloud.v2.src.resources.online_devices.online_devices.**OnlineDevices**(*get_client*)

**online**()

See devices online currently

**async online_async**()

See devices online currently

**class** formant.sdk.cloud.v2.src.resources.presence.presence.**Presence**(*get_client*)

**count**(*interval_query*)

Tells you if data has been ingested within a certain time period

**async count_async**(*interval_query*)

Tells you if data has been ingested within a certain time period

**class** formant.sdk.cloud.v2.src.resources.queries.queries.**Queries**(*get_client*)

**query**(*query*, *app_id='formant/sdk'*)

Queries objects based on data types

**async query_async**(*query*, *app_id='formant/sdk'*)

Queries objects based on data types

**class** formant.sdk.cloud.v2.src.resources.stream_current.stream_current.**StreamCurrent**(*get_client*)

**query**(*scope_filter*)

Gets you the current value of a stream that has been configured to cache the current value

**async query_async**(*scope_filter*)

Gets you the current value of a stream that has been configured to cache the current value

**class** formant.sdk.cloud.v2.src.resources.views.views.**Views**(*get_client*)

**get**(*device_id*)

Get a device layout

**get_all**()

List all device layouts

**async get_all_async**()

List all device layouts

**async get_async**(*device_id*)

Get a device layout

**patch**(*id*, *partial_view*)

Update a device layout

**async patch_async**(*id*, *partial_view*)

Update a device layout

# INDEX

## I



## L



## M



## O



## P

## Q



## R



## S

## U



## V