PreProcessing API v1.0.1

Common provisions

Terminology

The terminology of RFC 2119 (specifically **must**, **should**, **may** and their negatives) applies. The word **will**, when applied to the Service API, has the same meaning as **must**.

Protocol

The API supports communication over HTTPS only.

Encoding

The API supports communication using JSON encoding only, with rare specific exceptions. The client **must** submit the headers `Content-Type: application/json` and `Accept: application/json` for all requests. Failure to do so **will** result in a `415 Unsupported Media Type` response. Unless otherwise specified, the API **will** include the header `Content-Type: application/json` with its response.

Authentication

The API is authenticated by a JWT bearer token. Two token sources are accepted: * Tokens generated by Amazon Cognito and acquired from the Hardware `accessory/login` endpoint;

The client **must** submit the header `Authorization: <JWT>` with all requests. Failure to do so, or submitting an invalid or expired JWT, **will** result in a `401 Unauthorized` response.

General responses

In addition to the AWS API Gateway responses and the specific responses for each endpoint, the server **may** respond with one of the following HTTP responses:

- `400 Bad Request` with `Status` header equal to `InvalidSchema`, if the JSON body of the request does not match the requirements of the endpoint.
- `404 Unknown` with `Status` header equal to `UnknownEndpoint`, if an invalid endpoint was requested.

Schema

Simple

The following simple types **may** be used in responses: * `string`, `number`: as defined in the JSON Schema standard. * `Uuid`: a `string` matching the regular expression `^[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}$`, that is, the string representation of an RFC 4122 UUID. * `Datetime`: a `string` matching the regular expression `/\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}(Z|+\d{2}:\d{2})/` and representing a date and time in full ISO 8601

format. * `MacAddress`: a `string` matching the regular expression `/{[0-9a-f]{2}(:[0-9a-f]{2}){5}/`, that is six groups of two hexadecimal characters, separated by colons.

Session

A `Session` object **will** have the following schema:

```
{
    "session_id": Uuid,
    "event_date": Datetime,
    "created_date": Datetime,
    "updated_date": Datetime,
    "session_status": string
}
```

The following constraints **will** apply:

- `session_status` **will** be one of the strings `CREATE_COMPLETE`, `UPLOAD_IN_PROGRESS`, `UPLOAD_COMPLETE`, `PROCESSING_IN_PROGRESS`, `PROCESSING_COMPLETE` or `PROCESSING_FAILED`. For a newly-created session, the status **should** be `CREATE_COMPLETE`.

Endpoints

Session

Create

This endpoint can be called to register a new session.

Query String

The client **must** submit a request to the endpoint `/session`.

Request

The client **must** submit a request body containing a JSON object with the following schema:

```
{
    "sensors": [ MacAddress, MacAddress, MacAddress ],
    "event_date": Datetime,
    "end_date": Datetime
}
```

- `sensors` **must** be a list of exactly three MacAddresses, each of which is the ID of a sensor from which data is being collected.
- `event_date` **should** reflect the time that the practice session began.
- `end_date` **should** reflect the time that the practice session ended.

```
POST /preprocessing/1_0/session HTTP/1.1
Host: api.env.fathomai.com
Content-Type: application/json
Authorization: eyJraWQ...ajBc4VQ

{
    "sensors": [
        "11:22:33:44:55:66",
        "22:33:44:55:66:77",
        "55:22:33:44:55:66"
    ],
    "event_date": "2016-12-09T08:21:15.123Z"
}
```

Responses

If the registration was successful, the Service **will** respond with HTTP Status `200 Ok` or `201 Created`, with a body with the following syntax:

```
{
    "session": Session
}
```

Upload data

This endpoint can be called to upload data for the session.

Query String

The client **must** submit a request to the endpoint `/session/<session_id>/upload`, where `session_id` **must** be a Uuid of a previously-created session.

Request

Exceptionally, this endpoint accepts data in binary format instead of JSON. The client **must** submit a `Content-Type` header of `application/octet-stream`, and binary data in the body of the request.

```
POST /preprocessing/1_0/session/92d694eb-3d53-46fa-8b14-746c9b5380ef/upload
HTTP/1.1
Host: apis.env.fathomai.com
Content-Type: application/octet-stream
Authorization: eyJraWQ...ajBc4VQ

[raw bytes]
```

The size of the request body **must not** exceed 8MB.

Clients **should not** repeat upload requests which have previously succeeded, but **may** retry

requests for which an error response, or no response at all, was received.

Clients **must not** attempt an upload request for a session which has previously been successfully [marked as upload completed](#Completing data upload).

Responses

If the upload was successful, the Service **will** respond with HTTP Status `200 OK`, and with a body with the following syntax:

```
{
    "session": Session
}
```

If the request was not successful, the Service **may** respond with:

- `406 Not Acceptable`, with a `Status` header equal to `InvalidContent`, if the body of the request was not correctly encoded.

Completing data upload

Once all the data from a session has been uploaded, the client **should** use this endpoint to mark the upload as complete; the Service **may** then begin processing and analysing the data.

Query String

The client **must** submit a request to the endpoint `/session/<session_id>`, where `session_id` **must** be the UUID of a previously-created session. The HTTP method **must** be `PATCH`.

Request

The client **must** submit a request body containing a JSON object with the following schema:

```
{
    "session_status": "UPLOAD_COMPLETE"
}
```

- `session_status` **must** be the string `UPLOAD_COMPLETE`.

```
PATCH /preprocessing/1_0/session/92d694eb-3d53-46fa-8b14-746c9b5380ef HTTP/1.1
Host: apis.env.fathomai.com
Content-Type: application/json
Authorization: eyJraWQ...ajBc4VQ

{
    "session_status": "UPLOAD_COMPLETE"
}
```

Response

If the request was successful, the Service **will** respond with HTTP Status `200 OK`, and with a body with the following syntax:

```
{
    "session": Session
}
```

Where the `session_status` field **will not** be `UPLOAD_IN_PROGRESS` or `CREATE_COMPLETE` (and **should** be `UPLOAD_COMPLETE`, unless processing of the file has already begun).

If the request was not successful, the Service **may** respond with:

- `400 Bad Request`, with a `Status` header equal to `NoData`, if the Service has not received any data uploads for this session.