SEPA IoT Device Data API Documentation

Overview

This temporary API provides access for Tekh Limited to SEPA IoT sensor data from multiple device types. It consists of two primary endpoints:

1. **Date Bounds Endpoint** – returns the available date range for a device.
2. **Data Fetch Endpoint** – returns time series data for a device starting at a given timestamp, with a maximum of two weeks per request.

The API infrastructure supports around **5000 concurrent requests**.

Endpoints

1. Date Bounds

URL:

`https://a8p8m605b5.execute-api.eu-west-2.amazonaws.com/sepa_iot_device_date_bounds`

Method: `GET`

Parameters: - `device` (required) — Device identifier (DeviceEUI) - `type` (optional) — Device type (e.g. `HydroRanger`, `Theta`, `Droplet`, `Echo`, `Hygro`).

Notes: - `Whilst the type parameter is optional it is required when requesting data for HydroRanger or Theta devices.`

Example:

`https://a8p8m605b5.execute-api.eu-west-2.amazonaws.com/sepa_iot_device_date_bounds?device=70B3D549949EC862&type=Echo`

Response (200):

```
{
  "startTS": "2020-01-30T16:44:33.982+01:00",
  "endTS": "2025-09-09T06:53:16.623475834Z"
}
```

Notes: - `More recent data should all contain consistent timestamp formats but early devices may have mixed timestamp formats.`

2. Data Fetch

URL:

`https://oujshf1m2h.execute-api.eu-west-2.amazonaws.com/tekh_dataFetch`

Method: `GET`

Parameters: - `device` (required) — Device identifier (DeviceEUI). - `timestamp` (required) — Start time (ISO8601 UTC). Up to 14 days of data will be returned. - `type` (optional) — Device type (e.g. `HydroRanger`, `Theta`, `Droplet`, `Echo`, `Hygro`).

Notes: - `Whilst the type parameter is optional it is required when requesting data for HydroRanger or Theta devices.`

Examples:

HydroRanger:

`https://oujshf1m2h.execute-api.eu-west-2.amazonaws.com/tekh_dataFetch?device=F863663062792461&timestamp=2025-01-01T00:00:00Z&type=HydroRanger`

Response (200):


```
[
  {"DevEUI":"F863663062792461","Metadata":"{'ver': '1.0.9', 'battV': 3.34, 'signal': 18, 'iLevel': 0, 'iFlag': 0}","Payload":"039b","TimeStamp":"2025-01-01T00:00:00Z"},
  {"DevEUI":"F863663062792461","Metadata":"{'ver': '1.0.9', 'battV': 3.34, 'signal': 18, 'iLevel': 0, 'iFlag': 0}","Payload":"039f","TimeStamp":"2025-01-01T00:15:00Z"}
]
```


Theta:

`https://oujshf1m2h.execute-api.eu-west-2.amazonaws.com/tekh_dataFetch?device=F861275077961882&timestamp=2025-06-05T10:00:00Z&type=Theta`

Response (200):


```
[
  {"DevEUI":"F861275077961882","Metadata":"{'imei': 'F901405112736301', 'ver': '1.2.1', 'battV': 3.33, 'signal': 10, 'iLevel': 0, 'iFlag': 0, 'txTS': 1749123916}","Payload":"302b313837332e32352b31382e322b30","TimeStamp":"2025-06-05T10:00:00Z"},
  {"DevEUI":"F861275077961882","Metadata":"{'imei': 'F901405112736301', 'ver': '1.2.1', 'battV': 3.33, 'signal': 11, 'iLevel': 0, 'iFlag': 0, 'txTS': 1749124816}","Payload":"302b313837352e35312b31372e392b30","TimeStamp":"2025-
```

```
06-05T10:15:00Z"}
]
```


Generic:


```
https://oujshf1m2h.execute-api.eu-west-
2.amazonaws.com/tekh_dataFetch?device=2CF7F1203240000F&timestamp=2025-01-
01T00:00:00Z
```


Response (200):


```
[

{"Payload":"0101100c170000010210dc7c0100ebea","DevEUI":"2CF7F1203240000F","Ti
meStamp":"2025-01-01T00:02:25.227118861Z"},

{"Payload":"0101104416000010210a47d01002252","DevEUI":"2CF7F1203240000F","Ti
meStamp":"2025-01-01T00:18:16.032849045Z"}
]
```


Notes: - `Metadata` is often a string representation of a Python dict (single quotes). Parse it carefully — in Python use `ast.literal_eval(metadata_str)`, or convert single to double quotes if needed for JSON parsing. - Timestamps use the field name `TimeStamp` (capital S) and may include sub-second precision with a UTC `Z` suffix.

Usage Pattern: Full History Retrieval

1. **Get bounds:** call `/sepa_iot_device_date_bounds` to find `min_timestamp` and `max_timestamp`.
2. **Iterate:** call `/tekh_dataFetch` starting at `min_timestamp`. Each request returns up to 14 days of data.
3. **Advance:** use the last record's `TimeStamp` +1 second as the next `timestamp`.
4. **Repeat until** reaching `max_timestamp`.

Concurrency Guidelines

- API can handle ~**5000 concurrent requests**.
- Recommended: start with **50–200 parallel workers**, scale gradually.
- Implement retry with **exponential backoff** for 5xx/429 errors.
- Use **persistent connections** (HTTP keep-alive).

Data Parsing

Raw payloads returned by `/tekh_dataFetch` must be decoded using device-specific parsers.

Device Parsers

Defined in **Data-Parser-Examples.py**:

- **HydroRanger** — `parseHydroRangerPayload(payload, emptyDist)`
- **Theta** — `parseThetaPayload(payload)`
- **Echo** — `parseECHOdata(payload, emptyDist)`
- **Droplet** — `parseDROPLETdata(payload)`
- **Hygro** — `parseHYGROdata(payload)`

Device Metadata

Defined in **tekh_devices.json**: - Each device has a `DeviceEUI`, `DevName`, `SiteName`, and `EmptyDistance (HydroRanger and Echo only)`. - Use `EmptyDistance` for level calculations in HydroRanger and Echo devices.

Python Example

```python
import requests
from Data_Parser_Examples import parseHydroRangerPayload

BASE_BOUNDS = "https://a8p8m605b5.execute-api.eu-west-
2.amazonaws.com/sepa_iot_device_date_bounds"
BASE_FETCH = "https://oujshf1m2h.execute-api.eu-west-
2.amazonaws.com/tekh_dataFetch"

device = "F863663062792461"
type_hint = "HydroRanger"

# Step 1: get bounds
bounds = requests.get(BASE_BOUNDS, params={"device": device}).json()
min_ts = bounds["min_timestamp"]

# Step 2: fetch data
resp = requests.get(BASE_FETCH, params={"device": device, "timestamp":
min_ts, "type": type_hint})
data = resp.json()

# Step 3: parse payloads
empty_distance = 2236
for rec in data:
```

```
    parsed = parseHydroRangerPayload(rec["Payload"],
emptyDist=empty_distance)
    print(rec["TimeStamp"], parsed)
```

Best Practices

- Always use **UTC** timestamps (`Z` suffix).
- `Look up EmptyDistance` from tekh_devices.json it is required for water level calculation for HydroRanger and Echo devices.
- Use `type` query param, look up device type via `tekh_devices.json`.