Telemetry API Implementation guide

Versie: 1.9 05-03-2025

fudura

Version history

Version	Date	Change
1.0		Initial version
1.1	31-10-2022	Added the Shared-endpoints (for representatives)
1.2	26-04-2023	Corrected URL for developer-portal (from http to https)
1.3	28-04-2023	Added the GetSharedCustomerIds-endpoint (for representatives)
1.4	28-07-2023	Added information about the used timestamps
1.5	13-10-2023	Added customerId query parameter (for representatives)
1.6	30-10-2023	Styling
1.7	10-01-2024	Added the GetDetailedMeteringPoints-endpoint
1.8	08-01-2025	Added some context to the GetChanges endpoint and made sure the Dutch and English manual are consistent
1.9	05-03-2025	Added a small note to the GetChanges endpoint to clarify how the checkpoint works

fudura

Contents

Getting started

Introduction

The Telemetry API of Fudura provides access to telemetry and information about available telemetry streams. Aside from querying current telemetry, it is possible to stay up to date with new and updated telemetry with a change feed.

This document explains how to consume the API.

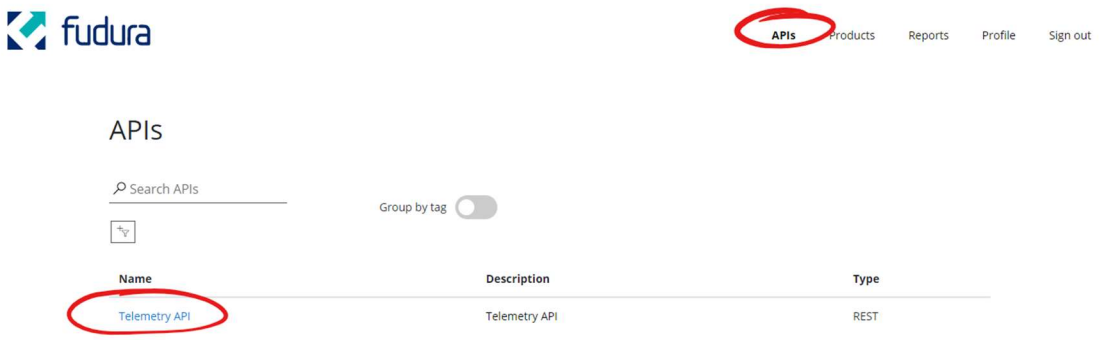
Developers Portal

To get started you need an account to Fudura's Developer Portal (https://developers.fudura.nl). This account can be obtained via Sales / Customer Support. An invite to create an account will be sent to the provided e-mail address.

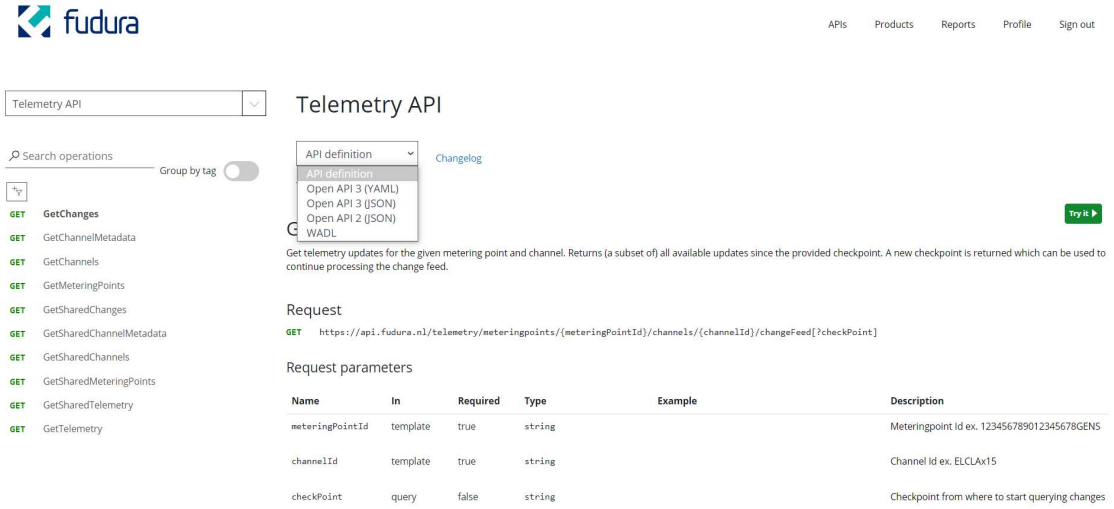
After creating a password and logging in to your newly created account you can access the specification for the Telemetry API, retrieve API keys and perform test request against the API.

Interactive API-Specification

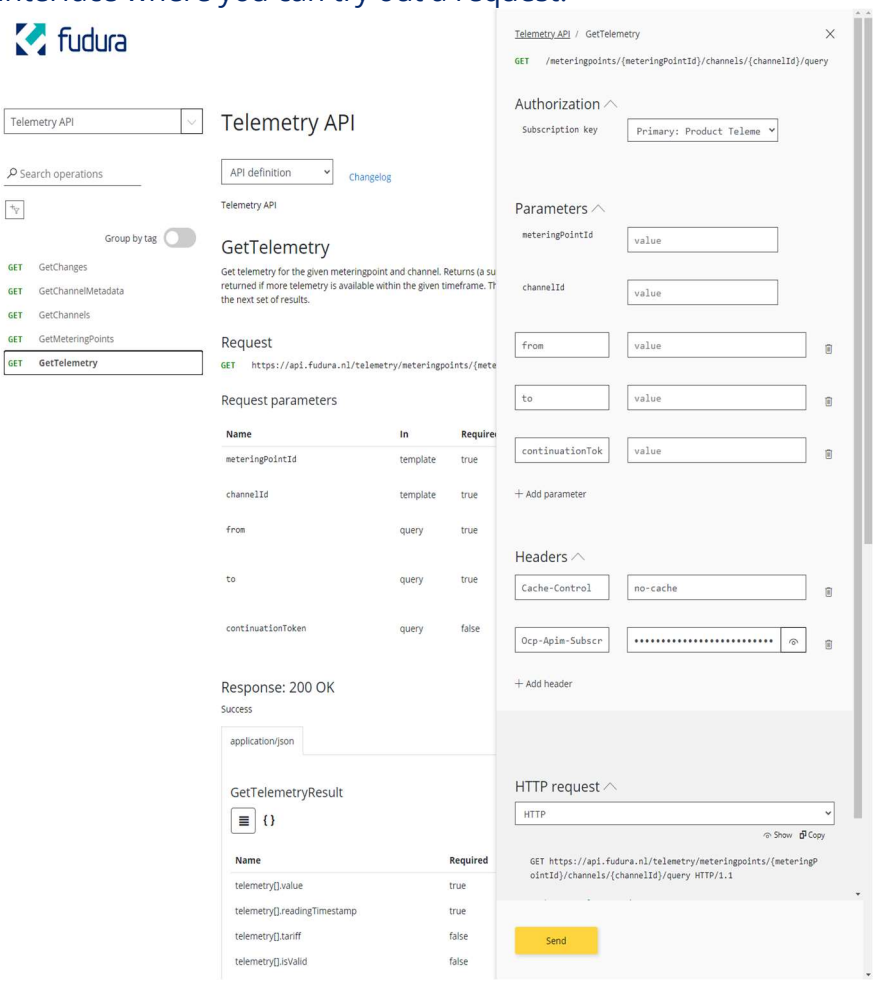
The specification for the Telemetry API can be found under "APIs" → "Telemetry API".



The specification is provided in both an interactive, graphical format as well as a downloadable document by selecting the desired format in the dropdown:

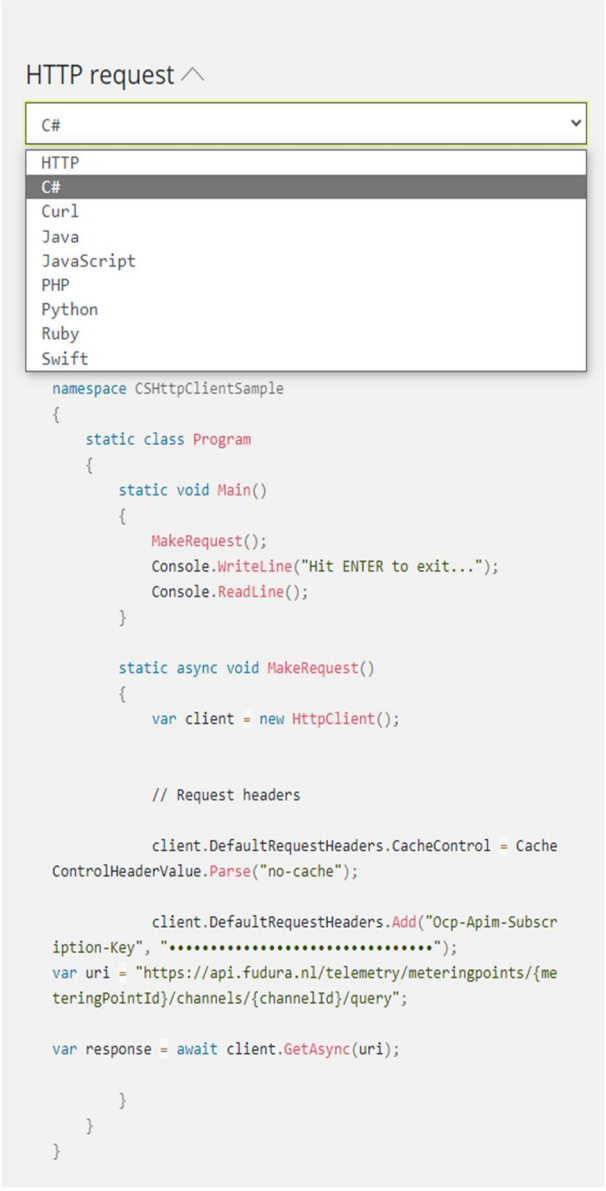


The graphical interface allows you not only to browse the endpoints, but also to send requests directly from the browser. The “try it” button (on the right) opens a new interface where you can try out a request.



After sending the request you get the options to reveal the secrets, or to copy the full request and used headers with just one click.

And by opening the dropdown at the HTTP request you can get examples on how to implement the request in code for the most popular programming languages.

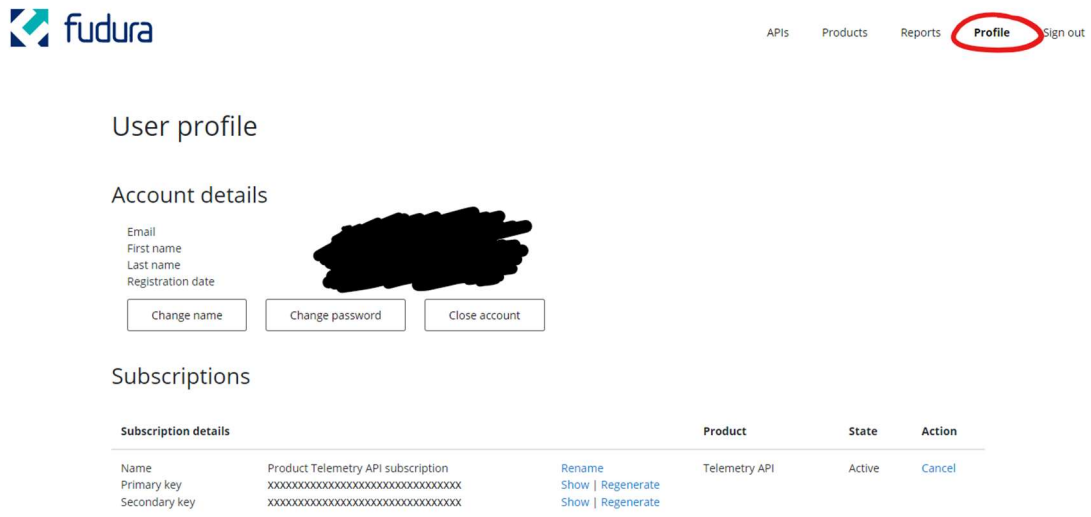


Timestamps

All times used in the Telemetry API are in UTC, in the “Zulu-time”-format. For each consumption reading the ReadingTimestamp represents the until-timestamp. For example: for the electricity consumption from `2023-07-25T15:45Z` till `2023-07-25T16:00Z` the ReadingTimestamp will be `2023-07-25T16:00Z`.

API Keys

Your API Keys can be found under “Profile” under “Subscriptions”. You can view and regenerate the keys here. **Note: Keep these keys in a safe place! Do not commit a key into version control or share it with others. If you suspect a key might be leaked, regenerate it, and update the key in your application.**



How to use

There are multiple ways to consume the Telemetry API, which option you should choose depends on your use case:

- Access to telemetry on an ad-hoc base → follow the instructions in chapter 2.
- Keep your systems up to date with updated and new telemetry → make use of the changefeed as described in chapter 3.

Telemetry requests

Intended Use

This chapter shows how to implement an integration with the Telemetry API to query telemetry. The method described below is tailored for on demand querying or use cases where only the current snapshot of telemetry is needed. If the candidate system needs to stay up to date with changes or new telemetry, make use of the change feed as described in chapter 3.

Implementation

To get started we need to understand these key concepts:

- **EAN** An 18-digit identifier for a (grid) connection.
- **Metering Point** Represents a measured connection.
- **Metering Point Id** An identifier for a metering point, consisting of an EAN optionally followed by a suffix.
- **Channel** Represents a channel of a single type of measurement.
- **Channel Id** An identifier for a channel.

Querying telemetry is done via the `/telemetry/meteringpoints/{meteringPointId}/channels/{channelId}/query` endpoint, but before we can query this endpoint, we need to decide which metering point and channel to query. The process of getting this information and finally querying the telemetry consists of the following steps:

1. Get Metering Points and Channels

The first step is to query the available metering points, including channels and authorization periods for which telemetry data is available. This is done using the `/telemetry/detailed-meteringpoints` endpoint, listed as operation **GetDetailedMeteringPoints** in the developer portal. This endpoint will provide information about which EANs are available to you, which metering points are available under each EAN, and for which time period(s) you are authorized to query the telemetry data.
When you are authorized to many metering points the response will be split. Each successive result can be requested by providing the given continuation token.

2. Get Channel Metadata (optional)

With the results from step 1 you have a list of metering points and channels you can query, but little context about the data that is contained in each channel. The `/telemetry/meteringpoints/{meteringPointId}/channels/{channelId}` endpoint, listed as operation **GetChannelMetadata** in the developer portal, can be used to get metadata about the channel. This includes the data type, measurement interval, description, first and last reading timestamp and more.

3. Get Telemetry

Now you have selected a metering point and channel you can finally query the telemetry with the `/telemetry/meteringpoints/{meteringPointId}/channels/{channelId}/query` endpoint, listed as operation **GetTelemetry** in the developer portal. When querying larger time ranges, only a subset of the telemetry available may be returned, in that case the results will also contain a continuation token. This continuation token is to be used to repeat the same query with the continuation token added as a query parameter. This will continue the query and return the next set of telemetry. Once no continuation token is returned the query is complete. The size of the result set is not guaranteed, but should be fairly consistent, so the amount of api calls needed to complete a query will correlate to the size of the queried time range and measurement interval of the queried channel.

Notes:

- The query endpoint requires a given date range that is within the authorization period for the given metering point, otherwise it will return an unauthorized result.
- The from and to parameters must be supplied as “Zulu” time following ISO 8601 (for example: `2022-07-15T13:20:05Z`).
- The from query parameter is exclusive.
- The to query parameter is inclusive.

Change feed requests

Intended Use

This chapter shows how to implement an integration with the Telemetry API which stays up to date with changes or new telemetry. For on demand querying read chapter 2 instead.

Implementation

To get started we need to understand these key concepts:

- **EAN** An 18-digit identifier for a (grid) connection.
- **Metering Point** Represents a measured connection.
- **Metering Point Id** An identifier for a metering point, consisting of an EAN optionally followed by a suffix.
- **Channel** Represents a channel of a single type of measurement.
- **Channel Id** An identifier for a channel.
- **Change Feed Checkpoint** A token that represents a single point in the sequence of changes in the change feed.

Querying changes is done via the `/telemetry/meteringpoints/{meteringPoin-tId}/channels/{channelId}/changes` endpoint. But before we can use this endpoint, we need to decide which metering point and channel to query and perform an initial load of telemetry. The process of getting this information and finally changes consists of the following steps:

1. Get Detailed Metering Points

The first step is to query the available metering points, including channels and authorization periods for which telemetry data is available. This is done using the `/telemetry/detailed-meteringpoints` endpoint, listed as operation **GetDetailedMeteringPoints** in the developer portal. This endpoint will provide information about which EANs are available to you, which metering points are available under each EAN, and for which time period(s) you are authorized to query the telemetry data.
When you are authorized to many metering points the response will be split. Each successive result can be requested by providing the given continuation token.

2. Get Channel Metadata (initial load)

With the results from step 1 you have a list of metering points and channels you can query, but little context about the data that is contained in a given channel. The `/telemetry/meteringpoints/{meteringPointId}/channels/{channelId}` endpoint, listed as operation **GetChannelMetadata** in the developer portal, can be used to get metadata about the channel. This includes the data type, measurement interval, description, first and last reading timestamp and more.

The result contains a property lastChangeFeedCheckpoint. The client application will need to remember this value and provide it with the first call to the **GetChanges** endpoint.

Attention: The received checkpoint is the latest known checkpoint for this meteringpoint/channel-combination. When you request data with the changefeed for this token immediately after getting it, it will most likely return an empty set, which means no new data is available yet for this meteringpoint/channel combination.

3. Get Telemetry (initial load)

Now you have selected a metering point and channel you can finally query the telemetry with the `/telemetry/meteringpoints/{meteringPointId}/channels/{channelId}/query` endpoint, listed as operation **GetTelemetry** in the developer portal. When querying larger time ranges, only a subset of the telemetry available may be returned, in that case the results will also contain a continuation token. This continuation token is to be used to repeat the same query with the continuation token added as a query parameter. This will continue the query and return the next set of telemetry. Once no continuation token is returned the query is complete. The size of the result set is not guaranteed, but should be fairly consistent, so the amount of api calls needed to complete a query will correlate to the size of the queried time range and measurement interval of the queried channel.

Notes:

- The query endpoint requires a given date range that is within the authorization period for the given metering point, otherwise it will return an unauthorized result.
- The from and to parameters must be supplied as “Zulu” time following ISO 8601 (for example: `2022-07-15T13:20:05Z`).
- The from query parameter is exclusive.
- The to query parameter is inclusive.

4. Get Changes

Now that you have retrieved the historical data, it is time to use the GetChanges endpoint to stay up to date with new telemetry. The GetChanges endpoint, `/meteringpoints/{meteringPointId}/channels/{channelId}/changeFeed={CheckPointId}`, can be called with the meteringPointId and channelId parameters. For the **first** call to this endpoint, use the checkpoint that we retrieved earlier through the GetChannelMetadata endpoint.

The response will contain all changes that were made to telemetry for this meteringPointId and channelId since the checkpoint. This includes both new and updated readings. The response will also include a property `checkPoint` if more data is available. More requests can be made to this endpoint until the `checkPoint` property is no longer present in the response. This means that we have caught up with the most recent telemetry. Make sure that your application remembers the last checkpoint value that was used in the request in order to continue from this checkpoint at a later time.

With changes we mean the following:

- New metering values
- Correction of previous metering values
- Changed validation result

The `GetChanges` endpoint only returns recent changes to telemetry. This means that:

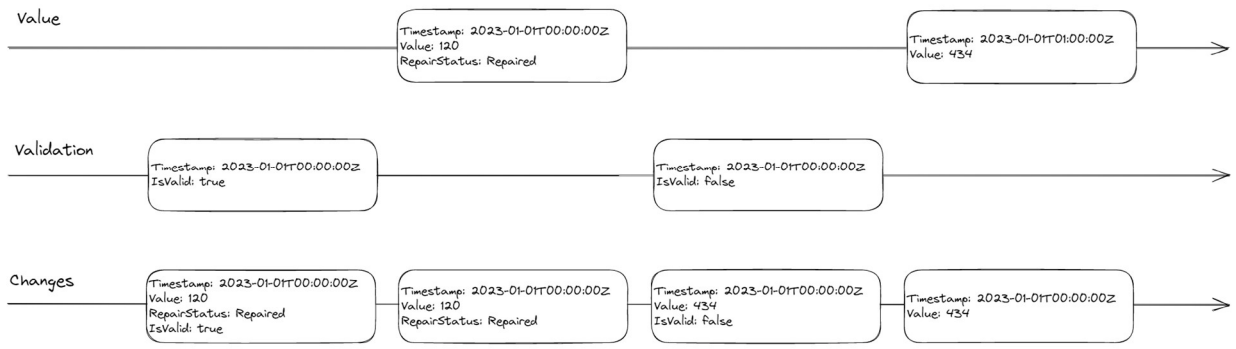
- Changes made more than 10 days ago will not be returned.
- Changes made less than 10 days ago *will* be returned, including changes regarding metering values from further in the past.

A checkpoint is only valid for a **limited period of 10 days**. After this period checkpoints are removed by the system. **No warning** is given when an invalid checkpoint is used. Instead, the response will contain data for the checkpoints that still exist for your telemetry data. Therefore, we advise to query the `GetChanges` endpoint at least daily, or more often if needed.

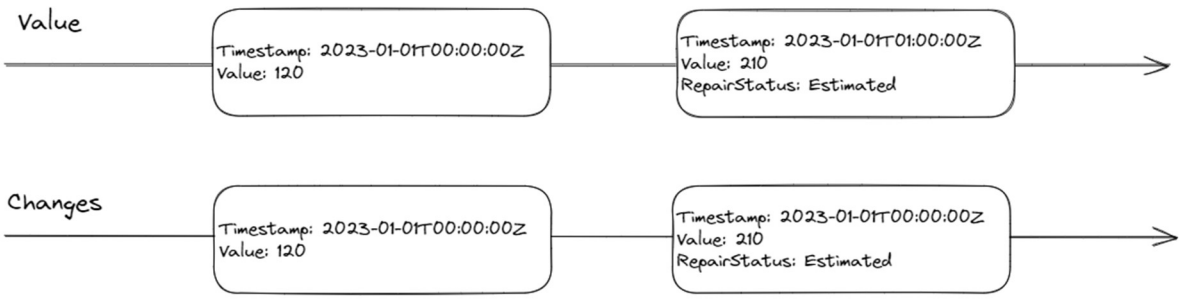
5. Processing Changes

The `GetChanges` endpoint represents a changefeed where all updates on telemetry for the given channel is recorded. This means that the results can contain multiple updates of the same reading (timestamp). All readings, or updates, in the result are ordered chronologically.

There are multiple causes for a timestamp to get multiple updates like corrections and validation. Since the validation process is asynchronous, the validation result will always be a later update to the reading. The following diagram shows a scenario where a single reading is collected and validated, after which a correction is made to that reading and is also validated.



Another example could be estimation, where the actual reading is later collected from the meter without validation.



To process the readings, every subsequent reading for a given timestamp should be considered a **full** update containing all available properties for that reading.

Request data as a representative

Introduction

A Fudura customer can authorize a representative to request the data on the customer's behalf. Once that approval is processed by the Fudura Customer Support, the authorized representative can request the data of the customer with the API account of the representative. There is no need to share API login credentials or authorization keys. The only thing the representative needs to know is the customerId (klantnummer).

Getting the customer ids

A representative can use the GetSharedCustomerIds endpoint to get a list of customer Ids of the customers for whom the representative is authorized.

Requesting data in name of a customer

All endpoints described in the previous chapters have an optional customerId-query-parameter. This parameter can be used as followed:

- Leave empty to get your own data.
- Use the customerId of the customer that mandated you to retrieve data on their behalf.
- Use a ***** to return all data you have access to, regardless of the customer.

Deprecated endpoints

Some endpoints are deprecated, but still available for backwards compatibility. Use the suggested alternatives to make sure your application still works with upcoming versions.

Deprecated endpoint	Alternative
GetSharedChanges	GetChanges + parameter customerId
GetSharedChannelMetadata	GetChannelMetadata + parameter customerId
GetSharedChannels	GetChannels + parameter customerId
GetSharedMeteringPoints	GetMeteringPoints + parameter customerId
GetSharedTelemetry	GetTelemetry + parameter customerId