

softserve

# **BENCHMARK 8760 PLATFORM TECHNICAL DOCUMENTATION**

For Jaros, Baum & Bolles, Inc.

Version 1.1

# Contents

<b>Executive Summary .....</b>	<b>2</b>
<b>1 Core Concepts .....</b>	<b>2</b>
<b>2 Links .....</b>	<b>2</b>
<b>3 Architecture .....</b>	<b>3</b>
<b>4 User Groups.....</b>	<b>5</b>
<b>5 Data Model for Configuration.....</b>	<b>5</b>
5.1 Model for Participants.....	6
5.2 Model for Meters.....	7
5.3 Model for Properties .....	8
<b>6 Data Model for Metered Data.....</b>	<b>10</b>
<b>7 Data Connection API .....</b>	<b>12</b>
7.1 Connector API .....	13
7.2 Data HTTP API.....	13
<b>8 Data Warehouse Model .....</b>	<b>14</b>
<b>9 Data Inspection Interface.....</b>	<b>18</b>
9.1 Scenarios for Data Inspection .....	18
9.2 Data Ingestion Portal.....	19
9.3 Data Analytics Portal .....	21
<b>10 Website .....</b>	<b>22</b>
<b>11 Connectors .....</b>	<b>23</b>
11.1 ConEd .....	23
11.2 Haystack.....	24
11.3 Other Connectors.....	24

# Executive Summary

This document describes the final technical architecture of the Benchmark 8760 Platform aimed at engineers who were not involved in the Platform development but intended to replicate the Platform or reuse it in other solutions.

Several additional considerations, discussions, and investigations were performed during the Platform development. They are described in project milestone memos and are not repeated in this document.

## 1 Core Concepts

The Benchmark 8760 Platform (or simply, the Platform) is designed to support the collection and analysis of metered data about commercial properties, such as electricity consumption, occupancy, CO<sub>2</sub> emissions, and ambient weather conditions. This data is combined in a data warehouse to allow making analytical queries and assess various scenarios of environmental performance benchmarking of the properties.

The **properties** refer to buildings, floors, individual rooms, or other indoor spaces. They simply identify the spaces that are subject to environmental data analysis and benchmarking that can be assessed through the metered data.

The **meters** refer to various sources of data about the properties and environment:

- Electricity meters monitoring energy consumption by the properties
- Occupancy meters monitoring utilization of the properties
- CO<sub>2</sub> grid emissions attributable to the properties given grid characteristics.
- Ambient weather allowing users to assess heating or cooling needs of the properties, etc.

Metered data is made available through various mechanisms, such as REST APIs provided by utilities or meter operators, CSV file upload, or emails.

## 2 Links

The Platform is deployed at <https://benchmark8760.org> as a proof of concept collecting data from several selected participants.

Technical instructions for new participants willing to feed data to the Platform are provided at <https://benchmark8760.readthedocs.io>.

The source code of the Platform is available on GitHub at <https://github.com/benchmark8760>.

# 3 Architecture

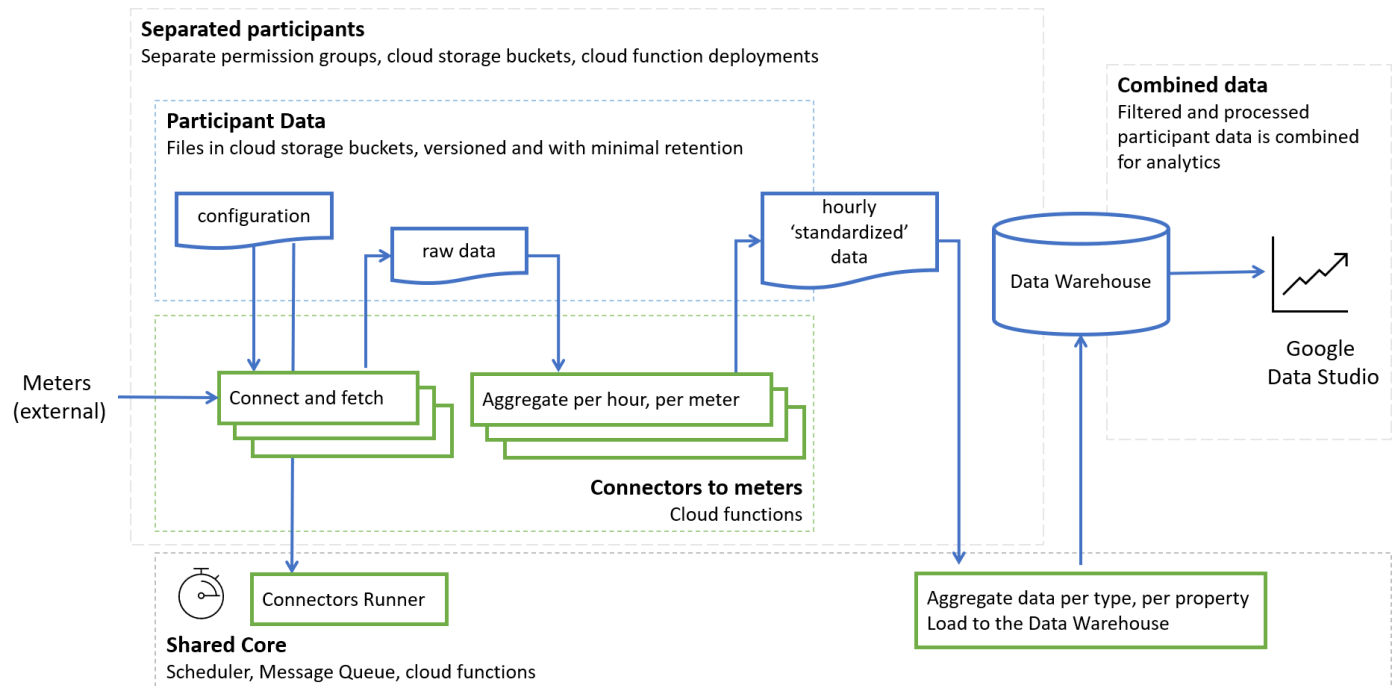
The Platform is architected as a cloud-native serverless solution built for Google Cloud Platform (GCP), relying on:

- Google Admin for all user and groups management
- Cloud storage buckets for all its storage
- Cloud functions for all computing needs
- BigQuery for the data warehouse
- Google Data Studio for visualization and analytics

The Platform serves participating organizations contributing data ('**participants**'), and these participants are isolated in the Platform, with each participant having:

- A special user group in Google Admin, holding users related to this participant
- A separate GCP Cloud storage bucket hosting configuration and data of that participant, with access restricted to the corresponding user group
- Data permissions set up in BigQuery to allow full access to (privileged) data of this participant only

The high-level architecture of the Platform is depicted in the figure below:



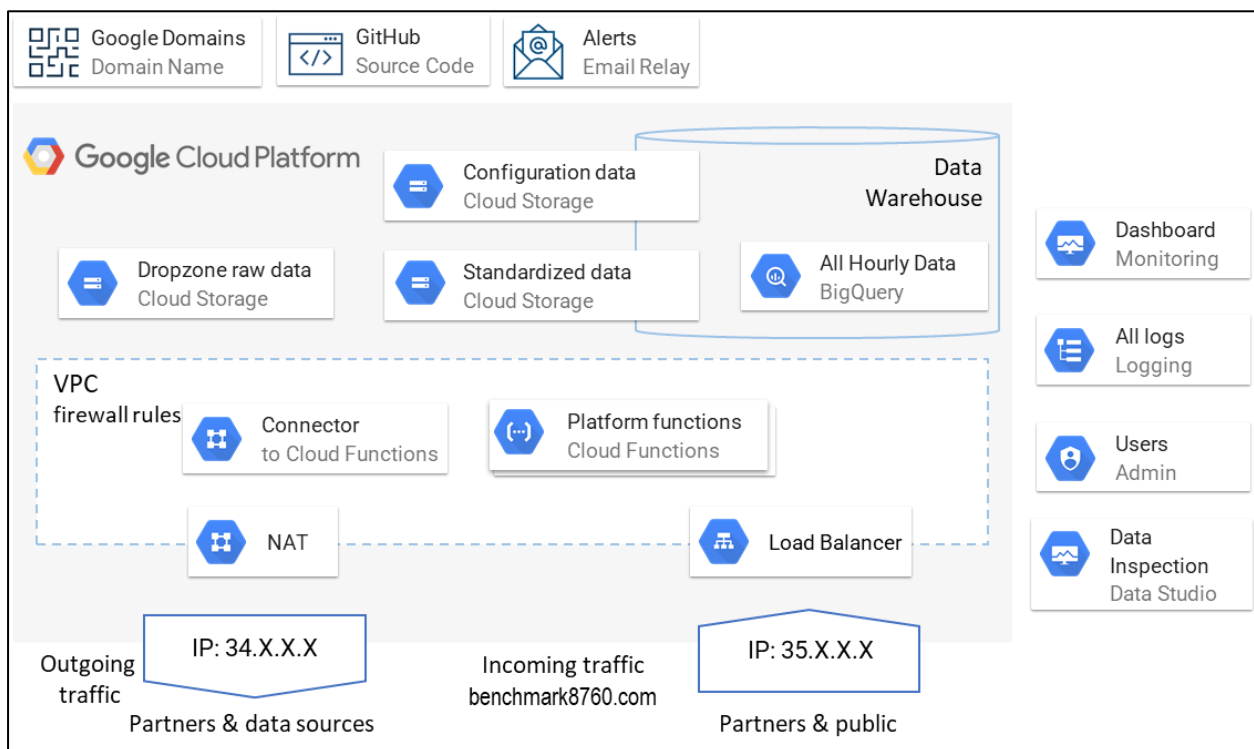
Every hour the Platform scheduler invokes the Platform **dispatcher**: a cloud function responsible for iterating through all participants, reading their configurations, and invoking the necessary connectors to fetch meter data.

The **connectors** are also implemented as cloud functions, parameterized, and invoked to fetch data for specific meters configured by specific participants. The connectors produce hourly data for each meter, keeping the raw data as fetched from the meter and performing all actions necessary to fetch, convert, aggregate, or interpret the raw data to obtain per-hour meter readings. These per-hour meter readings are called hourly 'standardized' data.

Then, the hourly data is further aggregated per data type to form the final hourly measurements of each data type that are then loaded into the data warehouse. In this way, for example, per-minute data coming from several occupancy sensors installed in different building areas are combined to estimate the total occupancy at each hour.

Platform users who belong to the participants are supported by a web configuration portal where they can examine meters configuration and status of data collection for the corresponding participant. Data warehouse data is made available for analytics through Google Data Studio.

From the cloud architecture perspective, the Platform looks like the following:



All Platform cloud functions are wrapped up in a VPC. There are two categories of cloud functions:

- Connectors that are invoked on schedule and do not accept any incoming connections. They pull data themselves. Some data providers require these pulls to come from predefined IP addresses

that are whitelisted. To provide that, a NAT component is deployed, and a static IP address is reserved (it is named '34.X.X.X' in the figure, referring to some specific static IP).

- Two small, specialized functions that support the web site and web portal and accept external connections. They are invoked through a load balancer and routing, responding to another static IP reserved for ingress traffic (it is named '35.X.X.X' in the figure, referring to some specific static IP).

The core Platform functionality is supported by several additional services: Google Domains, GitHub, GCP Alerts, and GCP logging tools.

The Platform does not use any VM or running server, no single Docker image, and no database. All permissions are managed with groups in Google Admin, including service accounts and permissions for human users.

## 4 User Groups

Platform participants are numbered in the Platform for technical convenience and to prevent unnecessary disclosure of their names. Participant numbers are incremental and are not reused during Platform operation. That means that if a participant leaves the Platform, then the infrastructure for this participant is scaled down. But the appropriate participant number is not taken by any other participant, and numbers of other participants are not changed.

For each participant, three groups are created in Google Admin ('X' refers to participant number):

- 'participantX\_admin' and 'participantX\_operator' holding human user accounts as members granting them read/write or read-only access to the data of the participant
- 'participantX\_services' holding service accounts allowed to process as members and granting them read/write access to the data in participant buckets
- 'participantX\_dw\_services' holding service accounts allowed to process as members and granting them necessary access to load and manage the data warehouse
- These groups are used to restrict access to users and services and to specific data elements of the data combined in the data warehouse described in section 8.

## 5 Data Model for Configuration

During the Platform design, it was required to align the data model used by the Platform with the data model used by the Energy Star Portfolio Manager (ESPM) and to extend it as needed. That

resulted in the XML Schema for all configuration and standardized data files used by the Platform accessible via <https://benchmark8760.org/ns/main.xsd>.

## 5.1 Model for Participants

Platform participants are represented with:

- Name, address, footage, contact person, and a few other common attributes
- Configuration of connectors fetching data of certain meters
- Properties for which the data is collected

Each participant is configured with an XML file named 'config/participant.xml' located in the participants' Cloud Storage bucket. This file looks like the following (fragment, XML tags are indented but not closed for brevity, unqualified tags are assumed to belong to the benchmark8760 namespace defined in the XML schema):

```
<participant>
  <name>Main Street Properties
  <contact>
    <espm:email>j.smith@ ...
    <espm:address address1="Main Str 5" city="New York" ...
  <connectors>
    <connector>
      <meterURI>participant_1/config/meter_main_electricity.xml
      <meterURI>participant_1/config/meter_secondary_electricity.xml
      <timezone>EST
      <function>connector_coned
      <fetchStrategy>
        <pull/>
      <parameter bm:property="password" bm:value="secret"/>
      <rawDataLocation
        bucket="participant_1" path="electra/raw/electric"/>
    <connector>
      <meterURI>participant_1/config/meter_occupancy.xml
      ...
  <properties>
    <propertyURI>participant_1/config/property_1.xml
    <propertyURI>participant_1/config/property_2.xml
```

In the fragment above, a participant named 'Main Street Properties' has two connectors:

- The first connector is implemented with a cloud function called 'connector\_coned' that receives a 'password' parameter with a 'secret' value. It fetches data for two electricity meters and stores them in the 'participant\_1' bucket, 'electra/ raw/electric' path.
- The second connector collects occupancy data, with its configuration skipped for brevity.

The participant has two properties configured in the Platform with identifiers 'participant\_1/config/property\_1.xml' and 'participant\_1/config/property\_2.xml'. They are configured in XML files with the same GCP Cloud Storage names.

All dates and times are converted to the UTC zone in the Platform and stored in UTC in the standardized XML data files with metered data. To facilitate this conversion, connectors are augmented with the 'timezone' configuration field. Commonly, meters are configured to UTC or another time zone, which is different from the time zone of their physical location. All data processing is performed with GCP cloud functions not associated with a specific server location and operate in UTC.

## 5.2 Model for Meters

The meters are described in separate XML files each, made according to the following format (illustration, XML tags are indented but not closed for brevity):

```
<meter>
  <meterURI>participant_1/config/meter_occupancy.xml
  <type>Occupancy
  <unitOfMeasure>persons
  <updateFrequency>Hourly
  <bm:meteredDataLocation
    bucket="participant_1"
    path="density/standardized/meter_occupancy" />
  <haystack:haystack>
    <haystack:id>P135
  <audit>
    <espm:createdBy>J. Smith
    <espm:createdDate>2021-11-05T10:34:12Z
```

Each meter is described with:

- Meter identifier, equal to the GCP path to the meter description XML file (in the above example, bucket 'participant\_1', file 'config/meter\_occupancy.xml')
- Meter type and unit of measure.

The following types and applicable units of measure are allowed:



Type of meters	Units of measure
Occupancy	persons
Average Grid Emissions, Marginal Grid Emissions	lbs CO <sub>2</sub> per kWh
Ambient Temperature, Ambient Dew Point	Fahrenheit, Kelvin, Celsius
Ambient Wind Speed	meters per second
Ambient Wind Direction	degrees
Ambient Humidity, Ambient Cloud Cover	%

- Meter update frequency, with most updated hourly. Data for some meters is not collected automatically but uploaded manually on a weekly or even monthly basis. Providing meter update frequency allows monitoring data feed health and raising appropriate alarms when data is delayed.<sup>1</sup>
- Location of metered data used by other cloud functions that need to access it.
- Haystack tags (large vocabulary of sub-tags from Project Haystack<sup>2</sup>).
- Metadata.

The meter description files are created by project participants manually and uploaded onto GCP buckets, either using GCP means or through the Platform configuration portal described in section 9.2.

## 5.3 Model for Properties

Meters are associated with properties to represent the fact that some meters apply to a property. Some meters are physically attached to properties (such as electricity meters), with some referring to other metered data (such as ambient temperature or CO<sub>2</sub> emission of a specific region within an electric grid). The meters associated with a property are configured in the property XML configuration file. Each property may have several meters associated with it:

---

<sup>1</sup> Update frequency occurs twice: in meter configurations and connector configurations.

<sup>2</sup> <https://project-haystack.org/>

For example, a property may be associated with:

- A ConEd meter for energy
- A few iES Mach submeters that need to be subtracted from the master meter
- Ambient temperature for New York
- Wind data from an on-premises weather station
- An energy meter from on-premises solar panels affecting net emissions of the property
- A fleet of 12 Density occupancy meters

Several meters of the same type may be combined to compute the representative values, for example:

- Summing up two meters when a property consumes energy from two sources
- Subtracting some meters when a property has a master meter and a submeter for another property, and the submeter needs to be excluded from the energy benchmarking calculations
- Estimating representative values when the occupancy sensor is installed to one of 3 property entrances, and its readings need to be tripled to be representative

Representative values for each property are computed as weighted sums of associated meters using the following formula:

$$\text{Representative value} = \text{SUM}(\text{weight}_i \times \text{value}_i)$$

where weights  $\text{weight}_i$  are configured for each meter-property association, metered values  $\text{value}_i$  are read from the meters, and index  $i$  indicates iteration.

Common examples for weights are as follows:

- Weight 0 is used to ignore the meter, used when the configured meter is not yet taken into operation
- Weight 1 is used to include the meter completely, used when the meter provides representative value for the property, such as with a single energy meter per property
- Weight -1 is used to subtract the meter completely, used when the property has a master meter included with weight 1 and a submeter measuring data for another property, excluded from the environment benchmarking calculations with weight -1
- Weight 3 is used when an occupancy sensor is installed on one of 3 property entrances, and its readings need to be triplicated to be representative of the whole building
- Weight 0.35 is used when the property is responsible for 35% of the metered consumption because of a business agreement.

This configuration is the key part of the property XML configuration files that looks like the following (illustration, XML tags are not closed for brevity):

```
<energyMeterAssociation>
  <meterURI>participant_1/master_meter_property_1
  <weight>1
</energyMeterAssociation>
<energyMeterAssociation>
  <meterURI>submeter_property_2
  <weight>-1
</energyMeterAssociation>
<occupancyMeterAssociation>
  <meterURI>#door_2_meter
  <weight>4
```

In the example above, a given property may have a master meter labeled 'Master Meter Property 1'. This meter may monitor an electrical service shared by both the first property and a second building under construction next door. The portion of the electricity used by the building under construction may be submetered by a meter labeled 'Sub Meter Property 2'. To compute the whole building energy use of the first property, you must subtract the electricity measured by the second building's submeter. To describe this logic in a way that can be understood by a program calculating total building energy use, the first building's master meter is given a weight of 1, the second building's submeter is given a weight of -1, and the two meters readings are added together.

An occupancy meter is also described in the example. In this example, the property has 4 equally used front doors, one of which is equipped with an occupancy meter. To estimate the overall property occupancy readings, these meter readings need to be multiplied by 4, represented with the weight of 4 in the XML file.

## 6 Data Model for Metered Data

Hourly 'standardized' data is stored per data point, with each data point stored in a separate XML file together with the reference to the description of the corresponding meter configuration, period, and meta-information, as illustrated in the following figure (illustration, XML tags are not closed for brevity):

```
<meteredData>
  <meterURI>property_5/electricity_meter_3
  <startTime>2021-10-31T23:00:00
  <endTime>2021-11-01T00:00:00
  <usage>26.7
  <audit>
    <createdBy>ConEdConnector
    <createdDate>2021-10-31T23:15:25
```

These XML files are stored in cloud storage buckets of specific participants, and XML was selected as a data serialization format for metered data. This XML format is derived from the industry-standard ESPM data exchange format.<sup>3</sup>

---

<sup>3</sup> More compact serialization formats may be used for this storage, such as comma-separated formats or binary formats. They would result in reduced volumes of data stored in GCP cloud storage buckets and reduced compute costs due to the eliminated need to parse XML.

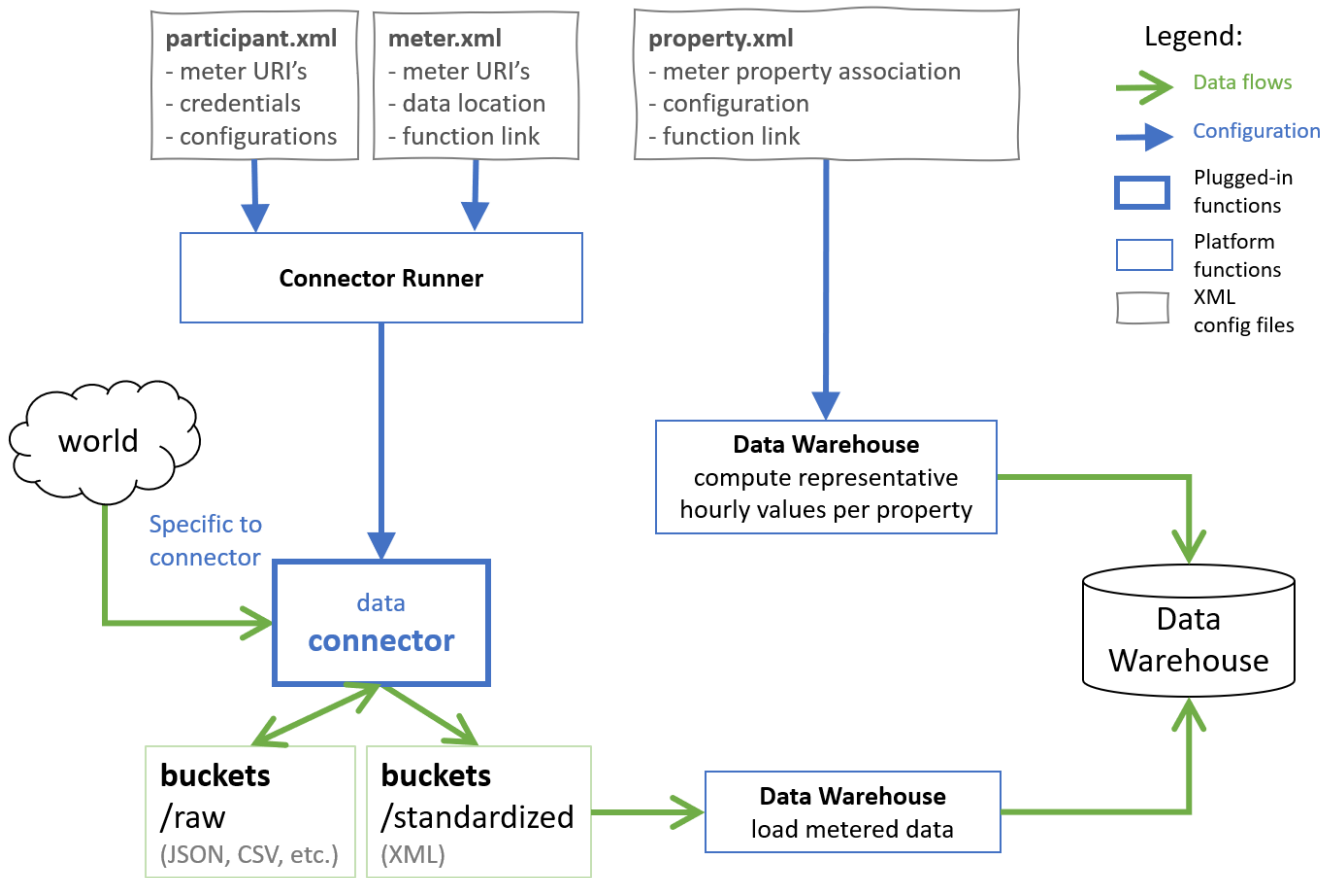
The benefits of using the XML format included:

- Ensuring the use of the proposed XML format, derived from ESPM
- Relying on GCP cloud storage REST API as the API for third parties willing to access the standardized data.

Given the business context of the Platform, it was decided that the benefits outweigh the costs (which are relatively low).

# 7 Data Connection API

The data processing flow invokes data connectors and data aggregators, as presented in the following figure:



The connectors collect the metered 'standardized' data loaded into the data warehouse together with the context data (such as property footage) and are aggregated to form per-property per-hour values.

## 7.1 Connector API

Connectors are deployed as GCP Cloud Functions inside the platforms GCP environment, but they are invoked using HTTP.<sup>4</sup> Each connector is made to respond to HTTP POST requests. On each invocation, it receives an XML `connectorRequest` message as the payload, as shown in the following example:

```
<connectorRequest>
  <parameter property="url" value="http://wattimte.com/api"/>
  <parameter property="password" value="secret"/>
  <rawDataLocation
    bucket="participant_1" path="/wattime/average/raw"/>
  <meteredDataLocation
    bucket="participant_0" path="/wattime/average/standardized"/>
</connectorRequest>
```

These parameters are made available to the connectors on each call.

Upon each invocation, the invoked connector returns the XML `connectorResponse` messages, as in the following example:

```
<connectorResponse>
  <recordCount>3</recordCount>
  <message>OK</message>
</connectorResponse>
```

The returned record count would normally be set to 1 for hourly data fetch. The returned value of 3 in the above example indicates that the connector fetched 3 records, possibly because of a previous downtime.

## 7.2 Data HTTP API

The data resides on GCP cloud storage buckets and can be accessed with the cloud storage REST API.<sup>5</sup> Recorded data of a meter may be accessed with an HTTP call to a specific file in a specific bucket at URL `'/b/BUCKET/o/OBJECT'`, for example:

```
/b/participant_1/coned/standardized/3/2021-11-08T14:00:00
```

that will refer to standardized readings of meter number 3 made on November 8, 2021, at 2 p.m. by a ConEd connector operated by participant number 1 of a consortium.

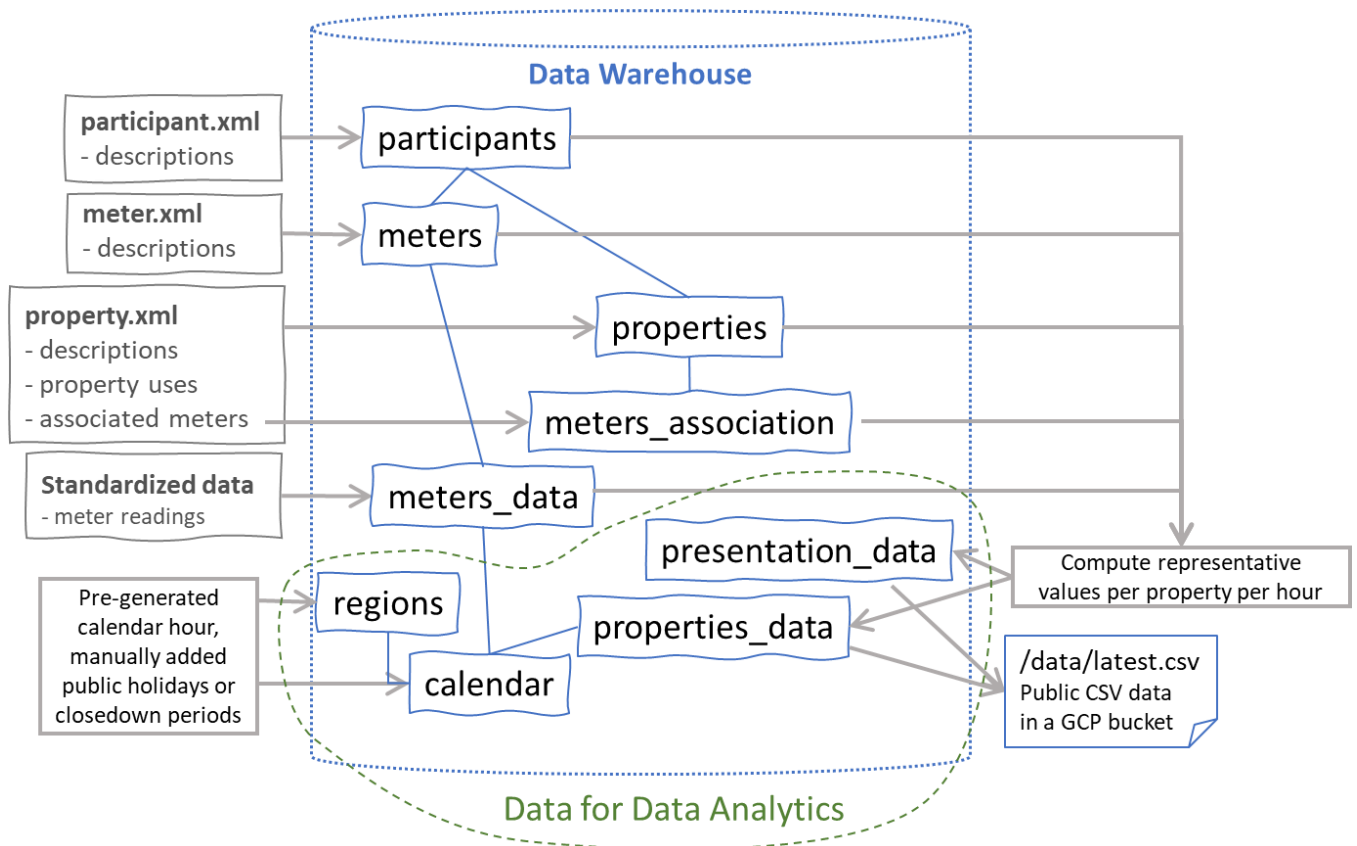
<sup>4</sup> External connectors that are deployed outside the platforms' GCP environment may be called using this approach as well.

<sup>5</sup> [https://cloud.google.com/storage/docs/json\\_api/v1](https://cloud.google.com/storage/docs/json_api/v1)

Uploading data this way should be avoided because it will intervene with internal data processing flows.


## 8 Data Warehouse Model

The data warehouse is designed to put the metered data in the context of characteristics of the properties, environment, and time. During each data upload to the data warehouse, the data from XML files located in GCP cloud storage buckets are extracted and used to populate several data warehouse tables, as illustrated by the following:




The data warehouse tables have the structure described below.


**Regions** that represent a geographical region are mostly used as an anchor to the calendar with the following fields:

Field name	Type	Mode	Collation	Policy Tags 	Description
region_id	INTEGER	REQUIRED		JBB_policy_tags_production : public	Region identifier, internal and unique
country	STRING	REQUIRED		JBB_policy_tags_production : public	Country
state	STRING	REQUIRED		JBB_policy_tags_production : public	State
timezone	STRING	REQUIRED		JBB_policy_tags_production : public	Timezone
description	STRING	REQUIRED		JBB_policy_tags_production : public	Description of the region

A **calendar** that represents calendars specific to regions is used to compute the working hours of that specific region. A calendar would contain enumerated regular working days, considering public holidays, and various closures due to weather, pandemics, festivals, and other reasons. The 'Calendar' table has the following fields:


Field name	Type	Mode	Collation	Policy Tags 	Description
ref_region_id	INTEGER	REQUIRED		Hidden policy tag	Reference to the region, regions here mostly differ in their working days schedule
hour_id	INTEGER	REQUIRED		Hidden policy tag	Hour identifier in the form of YYMMDDHH, such as 21113000 for the first hour of 30 Nov 21,
year	INTEGER	REQUIRED		Hidden policy tag	Year, such as 2021
day	INTEGER	REQUIRED		Hidden policy tag	Day of year, 1 - 365
date	INTEGER	REQUIRED		Hidden policy tag	Date in month, 1 - 31
month	INTEGER	REQUIRED		Hidden policy tag	Month number, 1 - 12
hour	INTEGER	REQUIRED		Hidden policy tag	Hour in a day, 0-23
dow	INTEGER	REQUIRED		Hidden policy tag	Day of week, Mon = 0, Sun = 6
working_day	BOOLEAN	REQUIRED		Hidden policy tag	Mon-Fri except public holidays
month_name	STRING	REQUIRED		Hidden policy tag	January-December
month_name_abbr	STRING	REQUIRED		Hidden policy tag	Jan-Dec
dow_name	STRING	REQUIRED		Hidden policy tag	Sunday-Saturday
dow_name_abbr	STRING	REQUIRED		Hidden policy tag	Sun-Sat

**Participants** that represent Platform participants with the following fields:


Field name	Type	Mode	Collation	Policy Tags 	Description
participant_id	INTEGER	REQUIRED		JBB_policy_tags_production : project	Participant number, also used in GCP permissions
name	STRING	REQUIRED		JBB_policy_tags_production : privileged	Participant name to show in the platform

**Properties** linked to the participants and described with their footage and other details, derived from the properties XML files described in section 5.3, with the following fields:




Field name	Type	Mode	Collation	Policy Tags 	Description
ref_region_id	INTEGER	REQUIRED		Hidden policy tag	Reference to the region where the property is located
property_id	INTEGER	REQUIRED		JBB_policy_tags_production : project	Property identifier for the use in DW, internal and unique
property_uri	STRING	REQUIRED		JBB_policy_tags_production : project	Property URI used in XML files
ref_participant_id	INTEGER	REQUIRED		JBB_policy_tags_production : project	Reference to participant
address1	STRING	REQUIRED		JBB_policy_tags_production : privileged	Building address
address2	STRING	REQUIRED		JBB_policy_tags_production : privileged	Building address for the apartment, suite, unit number, or other address designation
city	STRING	REQUIRED		JBB_policy_tags_production : privileged	Building city
country	STRING	REQUIRED		JBB_policy_tags_production : privileged	Building country
state	STRING	REQUIRED		JBB_policy_tags_production : privileged	Building state
postal_code	STRING	REQUIRED		JBB_policy_tags_production : privileged	Building postal code
footage	INTEGER	REQUIRED		JBB_policy_tags_production : privileged	Building footage, sq. ft.
footage_10_categories	STRING	REQUIRED		JBB_policy_tags_production : privileged	Building footage category, fine-grained, scale of ten
footage_3_categories	STRING	REQUIRED		JBB_policy_tags_production : privileged	Building footage category, coarse-grained, scale of three

**Meters association** to properties, representing the weights of the meters that were valid for a certain hour, with the following fields:

Field name	Type	Mode	Collation	Policy Tags 	Description
ref_hour_id	INTEGER	REQUIRED		JBB_policy_tags_production : privileged	Reference to hour
ref_participant_id	INTEGER	REQUIRED		JBB_policy_tags_production : privileged	Reference to participant
ref_meter_id	INTEGER	REQUIRED		JBB_policy_tags_production : privileged	Reference to meter
ref_property_id	INTEGER	REQUIRED		JBB_policy_tags_production : privileged	Reference to property that is associated with this meter
weight	FLOAT	REQUIRED		JBB_policy_tags_production : project	Meter weight this hour

**Meters** themselves derived from the meters XML files described in section 5.2, with the following fields:

Field name	Type	Mode	Collation	Policy Tags 	Description
meter_id	INTEGER	REQUIRED		JBB_policy_tags_production : project	Meter identifier for the use in DW, internal and unique
meter_uri	STRING	REQUIRED		JBB_policy_tags_production : project	Meter URI as it appears in XML documents
ref_participant_id	INTEGER	REQUIRED		JBB_policy_tags_production : project	Reference to participant
type	STRING	REQUIRED		JBB_policy_tags_production : privileged	Meter type
unitOfMeasure	STRING	REQUIRED		JBB_policy_tags_production : privileged	Unit of measure
updateFrequency	STRING	REQUIRED		JBB_policy_tags_production : privileged	Expected frequency of data updates
haystack	STRING	REQUIRED		JBB_policy_tags_production : privileged	Haystack tags

**Metered data** representing the standardized data as described in section 6, with the following fields:

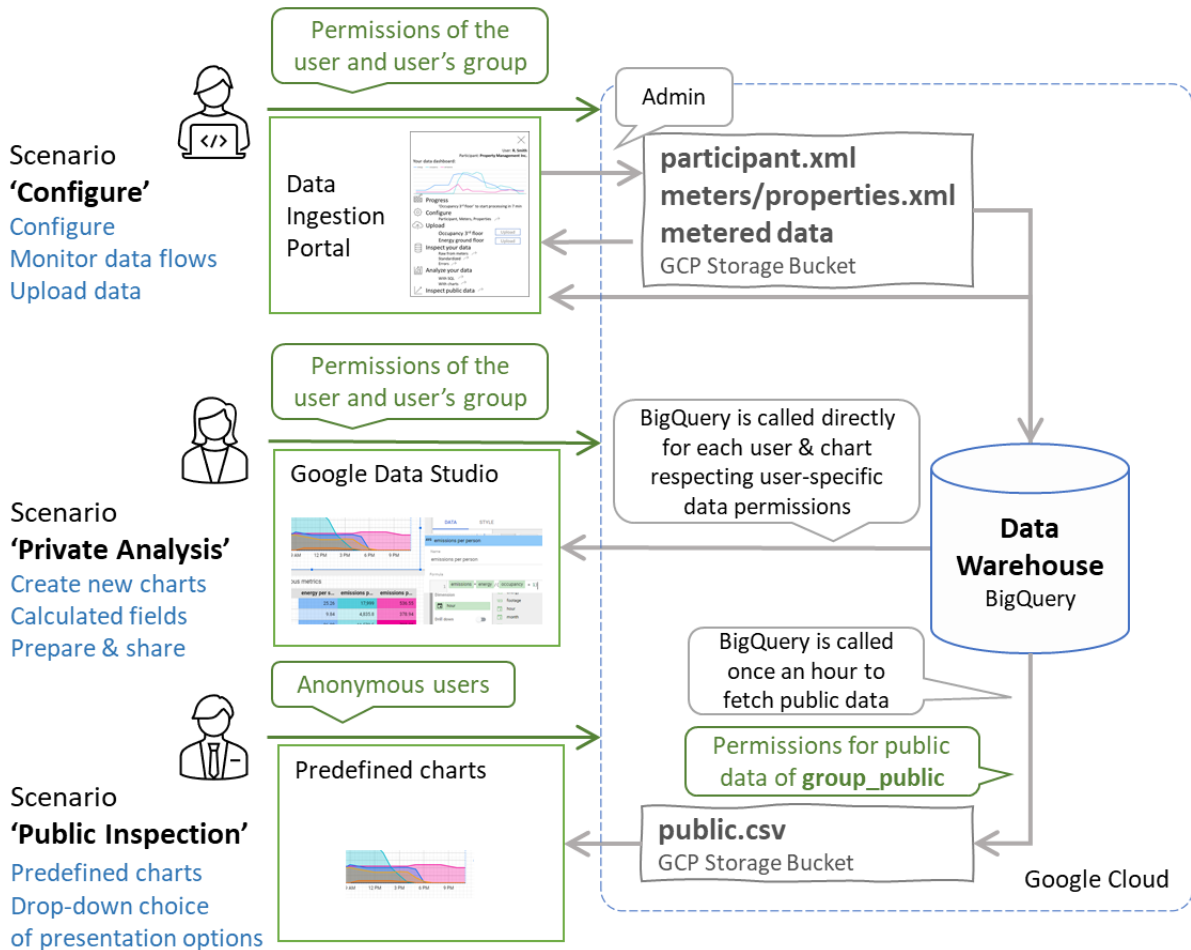
Field name	Type	Mode	Collation	Policy Tags 	Description
ref_hour_id	INTEGER	REQUIRED		JBB_policy_tags_production : privileged	Reference to hour
ref_participant_id	INTEGER	REQUIRED		JBB_policy_tags_production : privileged	Reference to participant
ref_meter_id	INTEGER	REQUIRED		JBB_policy_tags_production : privileged	Reference to meter
data	FLOAT	REQUIRED		JBB_policy_tags_production : privileged	Metered data, using units of measure of the meter

The data fields are subject to the BigQuery data protection mechanism represented with the policy tags.

# 9 Data Inspection Interface

## 9.1 Scenarios for Data Inspection

Three data inspection scenarios are supported by the Platform as shown in the following figure:



Users are served with three software components, one per scenario, that, in turn, interact with the files on the GCP storage buckets and the Data Warehouse. They are described in detail in the following sections.

## 9.2 Data Ingestion Portal

The Data Ingestion Portal provides Real Estate Participants with the ability to manage their data ingestion process. The Data Ingestion Portal is protected and is only available to the authenticated users explicitly added to the corresponding participant-specific user group.

The Data Ingestion Portal is designed as a single-page website providing the following features:

- Logging in and logging out
- Examining all meter configurations managed by the participant
- Examining the status of data collection by each meter
- Examining validation errors<sup>6</sup>
- Uploading data for the meters for which raw data is expected to be provided manually via HTTP upload
- Examining the configuration of properties managed by the participant
- Examining the status of data collection by each property
- Examining dashboards with collected data per-property
- Examining and uploading (for Administrators) participant, meter, and property configuration files

Platform participants are authenticated by GCP and are granted the read/write access to their storage buckets accordingly to their groups ('participantX\_operator' granting read access or 'participantX\_admin' granting read/write access, where X is the participant number in the platform). Generally, platform participants do not have permissions to access the GCP console, only access to their data. The Data Ingestion Portal is designed to perform GCP REST API queries to GCP cloud storage directly, requiring no special backend support except the group resolution function described in section 10.

All meters configured for a participant are shown together with their details and information about the time of the latest data ingestion, like the following (sample data):

Bucket prototype\_develop-epbp\_participant\_1, your role is Admin

**Access codes obtained from metered data providers** Download Participant XML Config  
No registered codes

**Status of data collection by each meter configured by the participant**

Meter URI	Type	Exp. updates	Last update	Latest Data File	Haystack
prototype_develop-epbp_participant_1/config/meter_density_1_occupancy.xml	Occupancy Hourly	2 hours ago		<div style="border: 1px solid gray; padding: 2px;">                     2022-03-15T12:00:00                      2022-03-15T13:00:00                      2022-03-15T14:00:00                      2022-03-15T15:00:00                 </div>	id: X <span style="float: right;">Download Meter XML Config Update Meter XML Config</span>
prototype_develop-epbp_participant_1/config/meter_density_2_occupancy.xml	Occupancy Hourly	2 hours ago		<div style="border: 1px solid gray; padding: 2px;">                     2022-03-15T12:00:00                      2022-03-15T13:00:00                      2022-03-15T14:00:00                      2022-03-15T15:00:00                 </div>	id: X <span style="float: right;">Download Meter XML Config Update Meter XML Config</span>

<sup>6</sup> The Data Ingestion Portal is expected to perform lightweight technical validation only, assisting human users responsible for conceptual validation of the configuration.



## 9.3 Data Analytics Portal

The Data Analytics Portal is based on the Google Data Studio,<sup>7</sup> a free tool provided by Google to perform open-ended data analysis and data-driven reports. It is integrated with the overall GCP security and data security system. Data Studio users are only allowed to access the data according to the BigQuery Data Warehouse data permissions, regardless of the type of exploration and charts they may be using.

The use case 'Private Analysis' is implemented by creating a few Data Studio data sources, allowing Data Studio self-enrolled users to explore data from the Platform's data warehouse based on BigQuery. The data sources are created by the Platform Administrator using the Data Studio interface that looks like the following:

The screenshot displays the Google Data Studio interface for a data source named "Hourly Building Data (Properties)". The interface includes a header with the data source name, a "Share" button, and a "Data freshness: 1 hour" indicator. Below the header, there are options for "Data credentials: Viewer", "Community visualizations access: Off", and "Field editing in reports: Off". A "CREATE REPORT" button is also visible. The main area shows a table of dimensions with columns for "Field", "Value", "Type", "Default Aggregation", and "Description". The dimensions listed are "average\_emissions", "electricity", "marginal\_emissions", and "occupancy". A callout box labeled "Permissions of the user would be applied" points to the "FILTER BY EMAIL" button.

Field	Value	Type	Default Aggregation	Description
average_emissions	123	Number	Sum	Average grid emissions this hour, as applicable to this building, lbs
electricity	123	Number	Sum	Average energy consumption during this hour, Watt
marginal_emissions	123	Number	Sum	Marginal grid emissions this hour, as applicable to this building, lbs
occupancy	123	Number	Sum	Average people count this hour, persons

The Platform Administrator may choose to develop several analytical data sources and share them as appropriate. It is anticipated that most would be shared with 'Anyone with the link can view' to let users analyze the data.

When a data source is shared with a user (with password), the user can create, modify, share, export, and publish analytical reports. The reports consist of (fragments of) web pages with:

- Text boxes with pre-defined text, freely allocated by report editors, using basic text formatting options
- Charts, freely allocated and configured by report editors, using a choice of chart types (lines, pies, etc.) with basic formatting options for chart axes, series, and background

<sup>7</sup> <https://datastudio.google.com/>

- Populating charts with the Data Warehouse fields, selecting certain fields to plot in data series, with sorting and filtering
- Creating dynamic fields computed from the Data Warehouse fields using basic arithmetic and statistical operations.

The overall text and chart generation user experience resembles a (lightweight) user experience similar to creating charts in Microsoft Office. Any other advanced analytics experience, such as that of Tableau or R or Machine Learning, is not provided in the project scope.

## 10 Website

The Platform has a website with two functions:

- Access point to the Data Inspection Interface implemented as a static website
- Landing point for web-based authentication flows

The web site serves the following endpoints:

<p><b>/</b> main (root) page</p> <p><b>/resources</b> static resources meant for being directly accessible by users, such a policies, T&amp;C, etc.</p> <p><b>/portal</b> data inspection and configuration portal</p> <p>    <b>/portal-resolve-participants</b> described below</p> <p><b>/data</b> raw data export location</p> <hr/> <p><b>/connect</b> connections to data providers:</p> <p>    <b>/1</b> connection to Con-Ed described in Section 11.1</p> <p>        <b>/redirect</b> callback or redirect authorization link for Con-Ed</p> <p>        <b>/notifications</b> receive notifications from Con-Ed</p> <p>        <b>/scope</b> scope selection for Con-Ed</p> <p>    <b>/2</b> another connection ...</p>	<p><i>Landing points for authentication flows</i></p>
--	---

All static resources are served from a GCP bucket, and all dynamic resources are served through a load balancer and GCP cloud functions.

Cloud function 'portal-resolve-participants' responds to the URL 'portal/portal-resolve-participants'. It receives the username and authorization key and returns a list of participants to which the user has access to, together with the associated user role ('operator' or 'admin').

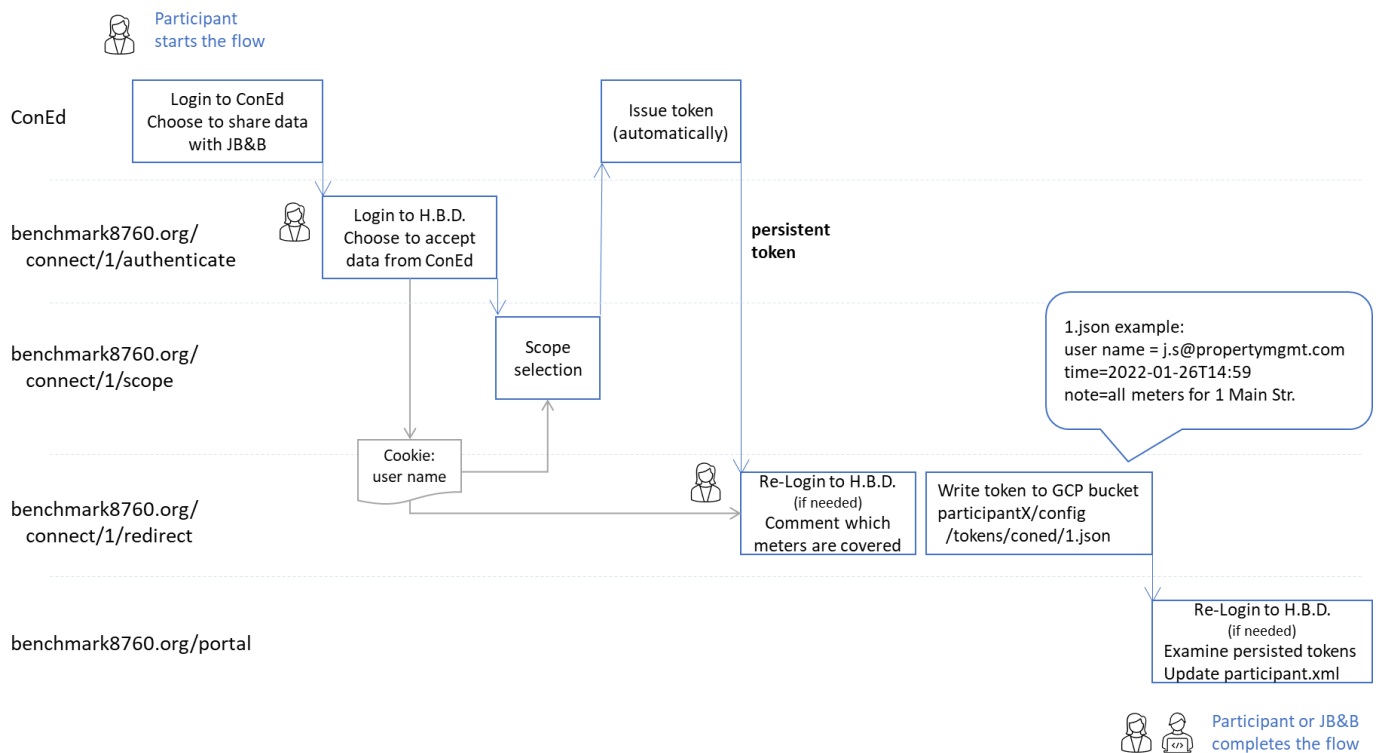
In GCP, permission to read all groups is needed to check group membership, and such permission cannot be granted in a granular way to allow a user to read only the groups where the user is a member. This cloud function is needed to resolve group membership in a controlled manner and avoid granting each Platform participant user rights to read all groups.

# 11 Connectors

Several cloud functions, called connectors, are developed to feed data from specific data sources to the Platform. Most of them are very specific to a participant, working around the way the data is represented in files and systems of that specific participant. Two connectors, ConEd and Haystack, have broader reusability potential and are described in this section.

## 11.1 ConEd

ConEd (also referenced as ConEdison) is an energy company providing an API to read electricity meters. The ConEd integration uses OAuth authorization flow involving both back-end and front-end, as depicted below:



The flow starts at ConEd, where a user, member, or participant logs in to permit data sharing with the Platform administrator. Then, the user is redirected to the platform to log into the platform and obtain persistent ConEd access tokens that confirm the data-sharing permission. These tokens are stored in the 'config/tokens' directory in GCP buckets of the participants administrated by the user. The tokens are then made available in the data ingestion portal to be manually included in the corresponding `property.xml` configuration files. This operation is expected to be performed once per ConEd account.



Data is collected on an hourly basis once authentication is completed.

## 11.2 Haystack

The Haystack Project<sup>8</sup> is an open-source suite of technologies for modeling IoT data, such as energy consumption and occupancy. The Platform contains a connector that allows reading data from data sources that support Haystack.

## 11.3 Other Connectors

Several other connectors were developed for this project including ones for:

- Emailed CSV data ingestion
- FTP sites
- HTTP push-based ingestion
- Miscellaneous RESTful APIs

For the source code please visit [GitHub Benchmark8760 Repository](#) and for further explanation please contact [support@benchmark8760.org](mailto:support@benchmark8760.org) to request further technical documentation.

---

<sup>8</sup> <https://project-haystack.org/>