Amperecloud Platform API Quickstart

Amperecloud Platform API Quickstart

Considerations

Alongside the prerequisites, we recommend that you take the following steps in order to successfully work with our API:

- **GraphQL Client:** Using a GraphQL Client such as *Apollo Studio, Altair GraphQL Client* or *GraphiQ*L will considerably accelerate your query development, due to their introspection and autocomplete features. After setting the authorization header, they will be able to fetch the documentation from our endpoints automatically.
- **Rate Limiting:** Our API is rate limited based on Fair Use policies and might deny high frequency, complex queries. Please structure your data acquisition in a meaningful way and use cache layers where necessary.
- **Change Notices:** Our API - alongside our app, is extended numerous times per week. Existing features are deemed stable (unless they are flagged as *Experimental* or *Deprecated* in our documentation), but can switch to a *Deprecated* state within one month after prior notice. Deprecated queries and fields can be removed at any time. Please reach out to support@ampere.cloud to be enrolled in an API change newsletter.
- **User Rights:** All requests to our API are running in the scope of the user that generated the key. If you have a company administrator's API key, you will be able to execute all functions that are available to them.

Endpoint & Authorization

The Endpoint of cloud.visions GraphQL API is available under the following address:

```
https://api.ampere.cloud/apollo-gateway
```

In order to authorize against the API, an authorization token will be needed, supplied in the header:

```
Authorization: Bearer <token>
```

This token will have to be supplied by every user that wants to connect to cloud.visions API. If you have a graphical user interface allowing the user to configure the communication, prompt him to enter his API Token.

Amperecloud Platform API Quickstart

Example Queries

Getting Available Facilities

Since a user can be member of multiple companies in cloud.vision, the best approach to get all facilities a user can access is as such:

```
query {
  getCompanies {
    id
    name
    facilities {
      name
      id
    }
  }
}
```

This will output a result as such. You will need the returned id of the facility to request its readings.

```
{
  "data": {
    "getCompanies": [
      {
        "id": "5f588d76d35baae5d63db732",
        "name": "ampere.cloud GmbH",
        "facilities": [
          {
            "name": "Test Facility",
            "id": "5f588d6d217ddfb7de05a92b"
          }
        ]
      }
    ]
  }
}
```

Amperecloud Platform API Quickstart

Getting Available Readings

In cloud.vision, data is both available as Raw Readings (e.g. gathered data from the facility) and Analysis (e.g. interpreted data, such as self consumption rate)

You can query the available readings and analysis by passing the id of the facility into the device query:

```
query {
 getFacilityById(facilityId: "5f588d6d217ddfb7de05a92b") {
    readingTypes
  }
}
```

This returns a result as such:

```
{
  "data": {
    "getFacilityById": {
      "readingTypes": [
        "Pac",
        "Pdc",
        "energy"
      ]
    }
  }
}
```

You can use the returned `readingTypes` to request the values for a certain timespan.

Amperecloud Platform API Quickstart

Requesting Readings and Analysis

Using the `facilityId` and the available `readingtypes` from above, you can request data using the `readings` field of the facility.

Readings

The `readings` field expects a `readingInput` field, which is formatted as such:

```
input type ReadingInput {
      readingResolution: Required - One of: RAW, DAILY, MONTHLY, YEARLY
      readingType: Required - one of the available readingTypes for the facility
      from: Optional - ISO Date String
      to: Optional - ISO Date String
      readingAggregation: Optional - One of: SUM, MAX, MIN
}
```

Amperecloud Platform API Quickstart

Example 1: Get all Pac readings of the facility for a day

```
query {
  getFacilityById(facilityId: "5f588d6d217ddfb7de05a92b") {
    readings(readingInput: {
      readingType: Pac
       readingResolution: DAILY
      readingAggregation: SUM
      from: "2020-09-08T00:00:00Z"
      to: "2020-09-08T23:59:59Z"
    }) {
      unit
      readings {
        date
        value
      }
    }
  }
}
```

Result

```
{
  "data": {
    "getFacilityById": {
      "readings": {
        "unit": "KILOWATTS",
        "readings": [
          {
            "date": "2020-09-09T09:10:40.874Z",
            "value": 0
          }
        ]
      }
    }
  }
}
```

Example 2: Get produced energy of the facility for a day

```
query {
  getFacilityById(facilityId: "5f588d6d217ddfb7de05a92b") {
    readings(readingInput: {
      readingType: Pac
       readingResolution: DAILY
      aggregationType: SUM
      from: "2020-09-01T00:00:00Z"
      to: "2020-09-30T23:59:59Z"
    }) {
      unit
      readings {
        date
        value
      }
    }
  }
}
```

Result

```
{
  "data": {
    "getFacilityById": {
      "readings": {
        "unit": "KILOWATTHOURS",
        "readings": [
          {
            "date": "2020-09-09T09:10:40.874Z",
            "value": 400
          }
        ]
      }
    }
  }
}
```

Next Steps

You have now learned how to access our API, how to request simple and more complex data. From here on, you will be able use the documentation and visual editors to compose your own queries.

Thanks to the power of GraphQL a single query to our API is enough to fetch all the data you need in one step. By merging both sections above, you can, for example, request both the facility's energy production and the irradiation of its sensors in a single query.

In general, all features of our software (both to read and to write) are exposed in our GraphQL API, with the exception of legacy use cases (that cover less than 3% of the API), which still run on a REST based API.

If you need support or are missing a functionality, please do not hesitate to contact your ampere.cloud representative!

API Setup

Step 1: Retrieve the API Key from the Platform

1. **Log into the platform**
   - Open your browser and go to the platform's login page.
   - Enter your credentials and log in.
2. **Open the Developer Console**
   - Right-click anywhere on the page.
   - Select **"Inspect"** (or press `F12`).
   - Switch to the **"Console"** tab.
3. **Retrieve the API Key**
   - Enter the following command into the console and press **Enter**:

     ```
     copy(JSON.parse(localStorage.auth).token)
     ```

   → If you get an error in the console, first run "allow pasting" in the console.

   - The API key will be automatically copied to your clipboard.
4. **Securely store the API key**
   - Save the key in a secure location, as it will be needed for configuration in Apollo Studio.

Step 2: Configure Apollo Studio for Query Building

1. **Open Apollo Studio**
   - Navigate to Apollo Studio Explorer.
2. **Set the API endpoint**
   - Enter the following in the **"GraphQL Endpoint"** field:

```
https://api.ampere.cloud/apollo-gateway
```

3. **Set WebSocket endpoint for subscriptions**
   - If subscriptions are required, use the following endpoint:

```
wss://api.ampere.cloud/apollo-gateway
```

4. **Authenticate using the API key**
   - Open the **"Headers"** tab in Apollo Studio.
   - Add the following authorization header:

```
  "Authorization": "Bearer YOUR_API_KEY"
}
```

   - Replace **"YOUR_API_KEY"** with the key you retrieved earlier.
5. **Test the connection**
   - Click on **"Schema"** in Apollo Studio to check if the schema loads successfully.
   - Run a simple test query:

```
query GetCompanies {
  getCompanies {
    name
    id
    facilities {
      name
      id
    }
  }
}
```

- If the query returns data, the setup is complete. ✅