Emporia Partner API Overview

The partner API allows Emporia's partners to query and manage the Emporia devices associated with to their partners' account. The API operates over gRPC, which is a protocol that allows clients to be written in a wide array of languages. Partners can use a wide variety of programming languages to access the API; Emporia offers sample clients in Java and Python.

The Partner API is not designed to report real-time data which is why it does not expose second-by-second energy usage. Emporia devices record energy usage for every second but that data is buffered on device and only reported to the Emporia cloud a few times an hour. To avoid seeing "partial bars" where the reported energy consumption is not yet complete it is suggested to not request data more recent than 30 minutes ago.

As of March 2023, we recommend that partners limit their API calls to less than 100 calls per hour. The API provides the ability to request data for all associated devices in a single call so this does not limit the number of devices a partner can monitor and manage via the API. A typical usage pattern might be:

- Request the list of associated devices hourly.
- Request minute, 15-minute or hourly data for some or all devices not more frequently than once a minute, being aware that requests for recent time periods may return partial bars.
- Control loads such as EV Chargers and Smart Outlets as needed.

Authentication

All API access is controlled using authorization tokens. The same email and password used to access Emporia's Partner Portal can be used to access the API.

Proto file

The partner_api2.proto file in the protos directory defines the API methods and data structures. Comments in the proto file describe the available methods and the returned data.

Sample Clients in Java and Python

The downloadable zip file includes sample clients written in Java and Python. You can provide your Partner Portal login (email and password) to these clients to fetch information about the devices associated to your partner account.

API Call Throttling

API call frequency, database usage and bandwidth limitations may be imposed to avoid misuse of the API by partners or erroneous code.

API Interface

The Partner API is only available via gRPC, there is not a REST version of the Emporia API.

Additional Functionality

As our relationships grow with our partners, we will continue to develop the feature set of our Partner API to meet the needs of our partners to allow them to manager their customers' devices. Feel free to reach out to `partner@emporiaenergy.com` with suggestions for improvements or additional functionality.

Migrating your usage of the Emporia Partner API from V1 to V2

In May 2022, Emporia released a V2 of the API which offers additional functionality for controlling EV Chargers and home batteries. As of March 2023, the V1 API is no longer available.

This section describes the differences between V1 and V2 and provides guidance on migrating your existing V1 code to the new V2 API.

Both versions use gRPC and the server makes the two different versions available on specific ports:

- V1 (no longer available): partner.emporiaenergy.com :: 50051
- V2: partner.emporiaenergy.com :: 50052

HTTPS support

The V2 API requires that clients use HTTPS to protect the partner credentials and the customer data being exchanged.

For Java, you should remove the .usePlaintext() from the ChannelBuilder so that gRPC will default to using SSL to secure the channel. For Python, you should switch from insecure_channel to secure_channel.

Error Handling

The V1 API had a ResultStatus enum that returned VALID on success and AUTH_EXPIRED on failure. The V2 API follows the guidelines and reports authentication errors by calling onError with a Status of UNAUTHENTICATED.

Rather than checking the returned result_status value, clients should handle the unauthenticated error by calling authenticate to get a valid auth_token.

Functional Changes

The V2 API provides additional behavior to list and control Emporia's EV Chargers and home Battery systems.

The API now offers a number of ListXX methods to return the specific state and settings applicable to one or more devices. There are also UpdateXX methods to control a device; e.g., turning on an Outlet or changing the charge rate on an EV Charger.

Some existing methods changed names to be consistent with the new behavior and some variables were renamed to be plural to match Google's guidelines. Specifically: 1) The Device message:

a. Now has an enum for DeviceModel rather than a String

b. No longer provides the last_app_connect_time

c. Provides the timezone that the user has chosen for the Device

d. No longer provides the zigbee_mac, that is now part of the UtilityConnect message

2) The DeviceUsageRequest now requires the caller to specify the scale of the usage data using the DataResolution enum rather than a string.

3) DeviceListRequest was renamed to ListDevicesRequest. This message is now used for many of the available methods, including ListOutlets, ListEVChargers and ListBatteries.

4) SetOutletStatus was replaced by UpdateOutlets and there are parallel methods for controlling Batteries and EVChargers.

5) The OutletStatusRequest was replaced by UpdateOutletsRequest and OutletStatus was renamed to OutletSettings to be consistent with the new EV Charger and Battery methods.