ZapCharger Pro/ZapCloud integration

ZapCloud 3.1, revision 2017.07.14

Table of Contents

Zaptec's ZapCharger Pro is a cloud connected EV charging station. Cloud connection allows multiple ZapCharger Pro's to work in collaboration to maximize efficiency. It also enables a range of other options like payment solutions and smart house integration.

Currently there are three ways to interact with and receive information from ZapCharger's that will be detailed in this document:

1. ZapCloud API
2. ZapCloud web hooks
3. ZapCharger message subscription

To enable the integration options discussed in this document, your ZapCharger's need to be connected to the internet. This can be done using Wi-Fi or PLC (see more details in the ZapCharger installation/user manuals). It also needs to be configured as part of an EV-charger installation in the ZapCloud Portal (https://portal.zaptec.com). The installation reflects the physical circuits and other limitations enforced for the group of chargers. When a charger is in use, this configuration and data received from other ZapCharger Pro's in the installation are used to continuously monitor, control, and optimize the installation.

ZapCloud API

The ZapCloud API contains methods to request information related to ZapCharger Pro's and their installations. It also contains methods to allow 3rd parties to adjust some runtime parameters. The API is REST based and uses OAuth for authentication.

Authentication

The API methods need to be called with a valid OAuth bearer token. This token is obtained by posting the following form-data to: https://api.zaptec.com/oauth/token:

- `grant_type` = `password`
- `username` = `{your username}`
- `password` = `{your password}`

For a successful request an `access_token` will be provided. This token needs to be provided through the Authorization header for any API requests:

```
Authorization: Bearer {access_token}
```

> Most methods in the API requires a user with owner permissions for the installation/charger. If you're not the owner of the installation, and need to use these methods, the current owner can elevate your permissions.

API documentation

The current API documentation is maintained at: https://api.zaptec.com/help/index.

The ZapCloud API documentation is interactive, and after logging in using the login field at the top of the web-page, API methods can be executed through the Swagger UI.

> Part of the API is marked as **deprecated**. Do not create integrations using any deprecated methods as these will be removed without warning. For the same reason, if you are already using deprecated method, it is required that you refactor your code as soon as possible.

For API methods returning charger state observations, see the list of supported observation/state IDs in chapter State observation reference.

ZapCloud web hooks

Web hooks are used for operations that require a 3rd party interaction before being allowed, i.e. before allowing a charge session to start, and after a charge session is completed. External authorization and 3rd party payment solutions can be implemented using these web hooks.

Configuration

An installation can be configured with one of these two charge authorization methods:

- **Internal authorization**
  Charge requests are authorized internally by ZapCloud. Charge is authorized if charge authorization is disabled or the current user has a *user* permission for the charger or the charger's installation
- **External authorization**
  Charge sessions are authorized by 3rd party using the configured web hooks

Web hook configuration is found in the installation details page when the installation is configured with external authorization. These configuration options are accessible by Zaptec Support and your installations service partner.

- **Authentication URL**
  This configuration option is used to configure a URL where ZapCloud will authenticate the web hook requests. The authentication URL must provide an OAuth bearer token. The provided access token will be added through an Authorization header when calling session start and end web hooks. Though authentication can be omitted, **it is highly recommended that your production web hooks are authenticated!**
- **Authentication payload**
  This string will be posted to the authentication URL as HTTP form data. Depending on your web hooks authentication implementation, this will normally have a format like: `grant_type=password&username={username}&password={password}`
- **Session start URL**
  The web hook URL to call before a session is authorized to start. Sessions can be denied to start, depending on the result of the request. If external authorization is enabled and this web hook is not provided, the installation will use ZapCloud internal authorization
- **Session end URL**
  The URL to call after a charge session is finished (vehicle is disconnected from the ZapCharger)

Web hooks

Session start

The session start web hook will be called when a user initiates a charge by connecting an EV to a ZapCharger. The hook will be called with information on what charger and installation is requesting charge, and an optionally scanned RFID token code to identify the user. The 3rd party web hook implementation must look up the user based on the provided RFID token, or by linking the user to charger using other options, e.g. through a 3rd party app, SMS or similar. If charge is authorized the 3rd party system must create and return a unique UUID[1] session identifier to authorize the session. This identifier will be used to reference the session in the ZapCloud database and in further communication. If the RFID token is not authorized the service should return a HTTP 401 response, and the charge request will be denied. Other failure situations should result in an appropriate **failure** status code (>=400) being returned.

The web hook will be polled every 10 seconds until one of these events occur:

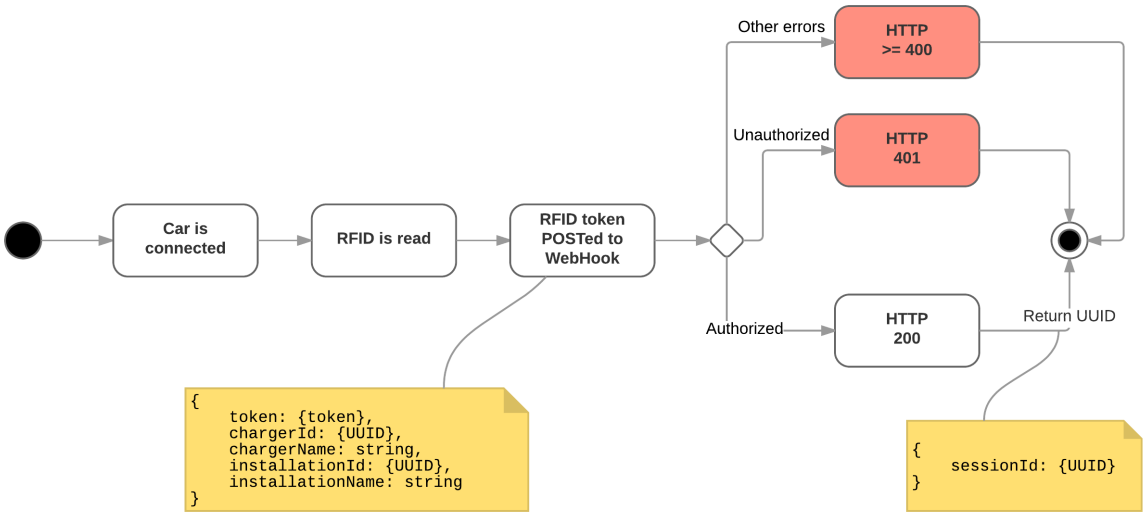
- Authorization is OK
- First failed authorization attempt with RFID token
- Poll timeout of 5 minutes have elapsed
- Charger is disconnected from vehicle

The user must scan RFID token after connecting to the charger. Because of this 3rd parties requiring RFID tokens will get session start requests without token code until this have been scanned. If RFID token is required, the 3rd part web hook must return 401 unauthorized for these calls.

There is no display on the ZapCharger and we have no way of informing about rejection-causes, other than flashing the LED Z-logo. Hence there is no need for the web hook to return user messages. A message can however be returned, and Zaptec will log these error messages for debugging purposes. For failed authorization attempts the charger Z-logo will flash red. 3rd parties authorizing users through their own apps, can provide more detailed messages in the app.

[1] https://en.wikipedia.org/wiki/Universally_unique_identifier

Web hook communication flow:



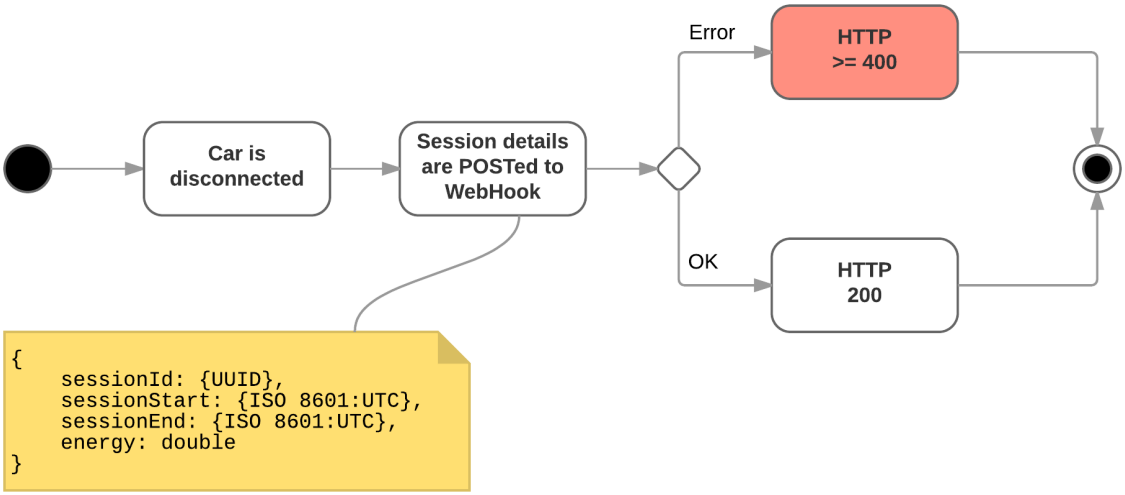
Session end

For pure authorization purposes, the charge start web hook is all that is required. If, in addition, payment should be processed after a charge session or a 3rd party needs to maintain a charge log, more information is required to be posted to the 3rd party system. This is done using the session end web hook.

After a charge session is completed (i.e. vehicle is disconnected), information about the session is posted to the configured web hook. The web hook will be provided the following details:

- **Session ID**
  This is the UUID created when authorizing the session, either by 3rd party or ZapCloud.
- **Session start time**
- **Session end time**
  Note: session ends when vehicle is disconnected, not when it was fully charged. The reason for this is that after a charge is initially completed, the car may at any point request more power, e.g. for cabin or battery heating.
- **Consumed energy**
  kWh

Web hook communication flow:



The web hook is called after the session have been closed in the Zaptec database. If the session close request fails (HTTP status >= 400), **no further attempts will be done to call the web hook**. Session status can at any point be requested from the API, see more details in section API: Session end.

For failures, we recommend that a detailed error message be returned in addition to the HTTP status code. This will be logged and better allow us to debug failure scenarios.

API: Session end

In addition to the above web hooks, an API method is provided to check the status of any given session. This can be used to check the status of a session, or request details of sessions whose session end requests have failed.

Sessions are referenced by their UUID session ID and the API method returns the same data as provided to the session end web hook.

More details about the session API can be found here:
https://api.zaptec.com/help/index#!/Session/Session_Get

Other details

- Serialization:
  - Dates are provided as ISO 8601 UTC strings: `2017-02-06T09:56:25Z`
  - UUID's are provided and expected as strings in the following format (.NET GUID): `123e4567-e89b-12d3-a456-426655440000`
- We do not expect users to be common across ZapCloud and the integrating party. The solution allows 3rd parties to integrate and authorize their own users, without any user synchronization between systems.
- For load balancing ZapCloud needs to know the topology of the installation, it's circuits and chargers. Even though, for authorization purposes, this may also need to be partly maintained in a 3rd party system, these details must exist in ZapCloud.
- Charge sessions authorized through 3rd parties will be created and stored in the ZapCloud charge session database. These sessions will be visible in ZapCloud as part of the statistics/charge history for the installation. Because we have no details of the users, **these sessions will be anonymous**. If there is a need for user specific statistics or logs, this will have to be provided by the 3rd party system.

ZapCharger messaging subscription

An installation may be configured with a message subscription. 3rd parties can connect to these subscriptions to get continuous push notifications when their chargers state change. State observations contain runtime details such as temperature, charge current, charge mode etc. A list is observations can be found in chapter State observation reference.

Messaging subscriptions must be activated per installation. This is done through the "messaging subscription" option under installation details in ZapCloud Portal. Currently this option is available for Zaptec Support and the installations service partner.

Enabling the subscription will configure an Azure Service Bus Topic for your installation. Messages received from installation chargers will be broadcast to this topic.

> There is a wide range of options available for receiving messages from Azure Service Bus: https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-dotnet-how-to-use-topics-subscriptions. In addition to using Microsoft's Service Bus libraries, it is also possible to consume messages using the standard protocol AMQP 1.0 (https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-amqp-overview).

Credentials

After the messaging subscription is enabled, and the Service Bus topic is created, connection details can be retrieved either:

- From the installation details in ZapCloud Portal; accessible by Zaptec Support and the installations service partner
- From the API method: https://api.zaptec.com/help/index#!/Installation/Installation_GetMessagingConnectionDetails, accessible by installation service partner and installation owners

Connection details can be combined as a Service Bus connection string:

```
Endpoint=sb://{Host}/;SharedAccessKeyName={UserName};SharedAccessKey={Password};EntityPath={Topic}
```

> The connection details are kept until the messaging subscription is disabled and the Service Bus Topic is removed. If you would like to reset your connection credentials to prevent unauthorized access to your installations messages: disabled messaging subscription and save, then enable the subscription and save again.

Message format

Messages are provided as JSON-serialized `ChargerState` objects (**https://api.zaptec.com/help/index#!/Charger/Charger_ChargerState**), containing:

- **`ChargerId`**
  The UUID of the charger providing the message
- **`StateId`**
  The ID of the changed state observation – list of supported state observation Id's can be found in chapter State observation
- **`Timestamp`**
  The UTC timestamp when the state observation was changed, provided as an ISO 86012 string
- **`ValueAsString`**
  The new state value, serialized as a string
  - Boolean: `true` = "`1`", `false` = "`0`"

[2] https://en.wikipedia.org/wiki/ISO_8601

State observation reference

Id	*Description*
-2	**IsOnline** • 1: charger is online • 0: charger is offline
201	**TemperatureInternal5** Internal temperature sensor 5 in degrees centigrade
202	**TemperatureInternal6** Internal temperature sensor 6 in degrees centigrade
270	**Humidity** Internal humidity in percent
501	**VoltagePhase1** Output voltage phase 1 in volts
502	**VoltagePhase2** Output voltage phase 2 in volts
503	**VoltagePhase3** Output voltage phase 3 in volts
507	**CurrentPhase1** Output current phase 1 in amperes
508	**CurrentPhase2** Output current phase 2 in amperes
509	**CurrentPhase3** Output current phase 3 in amperes
513	**TotalChargePower** Total instant charge power in watts
519	**SetPhases** • 1: TN phase 1 • 2: TN phase 2 • 3: TN phase 3 • 4: TN phase 1/2/3 • 8: IT phase 1 • 6: IT phase 2 • 5: IT phase 3
553	**TotalChargePowerSession** Total aggregated power for the current charge session in kWh
708	**ChargeCurrentSet** The allocated charge current for SetPhases in amperes
710	**ChargerOperationMode** • 1: no vehicle connected to ZapCharger • 2: vehicle connected; requesting to charge • 3: vehicle connected; charging • 4: vehicle connected; charging, changing current • 5: vehicle connected; finished
721	**SessionIdentifier**

The current session Id (GUID/UUID)

804 **Warnings**

809 **CommunicationSignalStrength**

- >= -70: reliable network connection (recommended) [3]
- < -70: minimal signal for connection, packet loss may occur
- < -80: unreliable connection, charger may randomly disconnect
- < -90: unusable

911 **SmartComputerSoftwareApplicationVersion**

Charger firmware version

[3] https://support.metageek.com/hc/en-us/articles/201955754-Understanding-WiFi-Signal-Strength

Common use cases

Authorization and payment solutions

Payment solutions, or solutions where an external system should authorize users, can be built using the ZapCloud web hooks discussed in chapter ZapCloud web hooks.

- The integration must expose two OAuth 2.0 authenticated HTTPS end points
    - **Session start**
      Will be called before charge is authorized. The 3rd party system decides whether the user is allowed to charge
    - **Session end**
      Called after a charge session has ended, with details on the charge session. Data can be used for calculating a cost for the charge session.
- *Recommended:* a scheduled task that periodically queries the session details API method for any sessions that have not been closed through the session end web hook. In case of network issues or 3rd party system downtime, the session end web hook may fail. The web hook will not be retried, and it is up to the 3rd party system to query and close any open sessions in their system

Dynamic load balancing

A ZapCharger's charging current and phase is dynamically controlled by ZapCloud. The optimal charge configuration is calculated based on the charger's installation properties and runtime state of other charger's is the installation. This is done transparently, always ensuring that as many chargers as possible is providing as much charging power as possible.

In some scenarios 3rd parties may want to limit the charge power available for an installation. E.g. to limit EV-charge power during costly peak hours, or prevent circuit breakers tripping in other high load scenarios. This can be done through setting `AvailableCurrent` using the installation update API method: https://api.zaptec.com/help/index#!/Installation/Installation_ExternalUpdate.

Dashboards

Using ZapCharger messaging subscriptions it is possible to build rich live dashboards and widgets that aggregate and present key metrics of one or more ZapCharger installations. Messages can be received using Azure Service Bus libraries or AMQP 1.0 and data can be integrated with a wide range of programming languages and solutions.

The live data can be used together with API methods like https://api.zaptec.com/help/index#!/Charger/Charger_ChargerState, that provide an instantaneous snapshot of the chargers current state, to provide a complete live representation of the installation and chargers.