COOLAUTOMATION
THE INTERNET OF CLIMATE (IoC)

CoolRemote Cloud REST API

Developers Guide

Early access version

Version 0.31
Apr 2020

Document Number: 0.31

Contract Number: info@coolautomation.com

Table of Contents

1. Scope

This document details the basic information required to work with the early access version of the CoolRemote Cloud REST API and SDK.

Note: early access version is subject to ongoing changes. Be aware this may require some changes to your implementation.

2. Overview

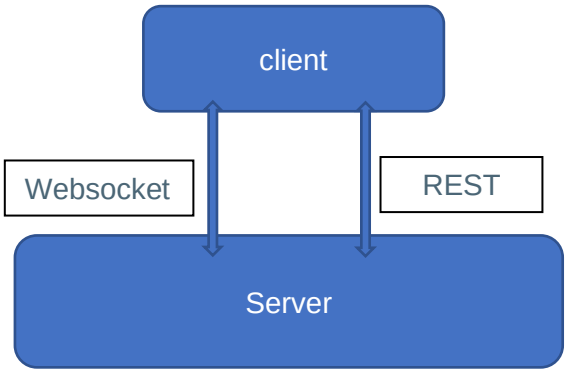
The CoolRemote Cloud is a PAAS intended to provide users and developers the ability to work with the CoolAutomation ecosystem.

2.1 Components

The platform includes:

- HTTP REST API
- Websocket interface
- A javascript SDK (future)
- A set of white-label web applications (future)
- A set of white-label mobile (android/IOS) applications (future)

2.2 Communication methods



The client is expected to communicate with the server via REST API and via a websocket.

The REST API is mainly for sending commands and querying the server, and the websocket is for receiving live status updates pushed from the server.

Using the websocket is not mandatory, as there are REST APIs that can provide the same functionality by polling for data but is highly recommended.

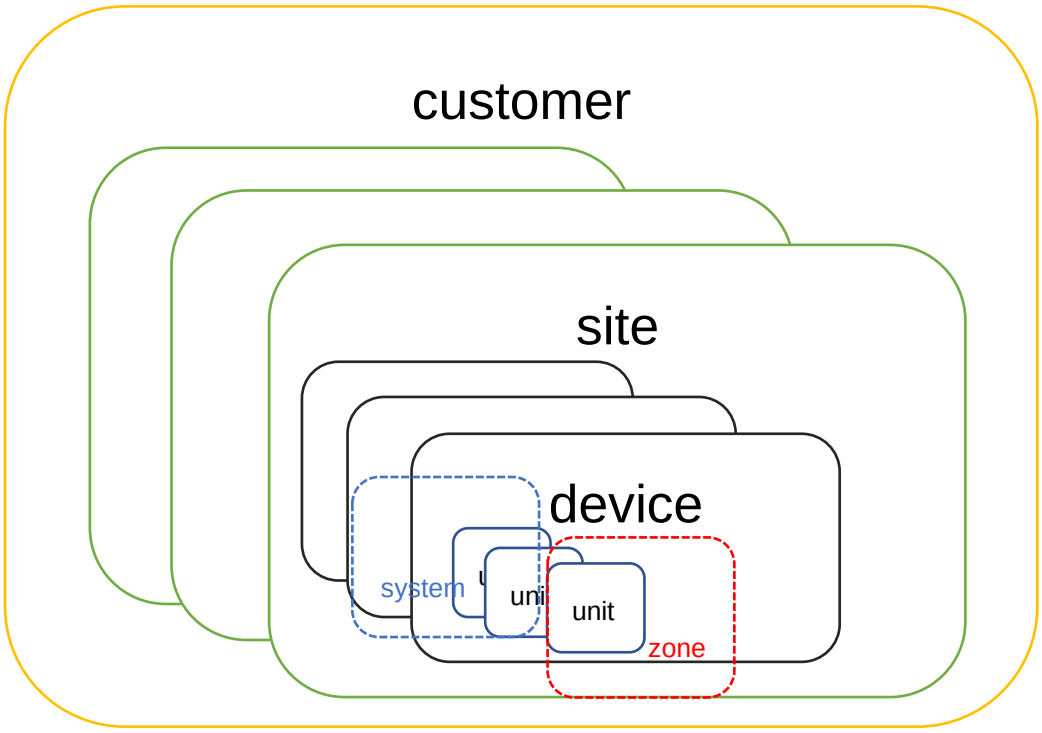
3. REST API

3.1 Overview

The REST API provides access to all the CoolRemote Cloud functionality via a REST-based HTTP API.

3.2 Entities

This section describes the hierarchy and relationship between logical entities in the platform.



3.2.1 Customer

The customer is the top hierarchy level exposed to an external user of our system.

It serves as a logical wrapper around all other customer specific entities.

3.2.2 Site

A site is a logical entity intended to represent a physical location where equipment is installed.

Each customer may hold multiple sites.

3.2.3 Device

A device is a logical entity intended to represent an actual physical CoolAutomation web-enabled device, such as the CoolMasterNet.

Each site may hold multiple devices.

3.2.4 Unit

A unit is a logical entity intended to represent a physical unit connected to a device, such as an AC unit / water heater / fan etc.

Units can not be added/removed by a user, but instead are detected by the device, once added.

Each device may hold multiple units.

3.2.5 Zone

A zone is a logical entity intended to provide easier control over a site. It lets the user control 1 or multiple units that are co-located in a unified manner.

For example – if a site has 2 AC units located in the same room, the user might want to create a zone named after this room and add the 2 units to it. Now it can control the temperature of these 2 units from a single control.

Units can be added and removed from a zone.

A site may hold multiple zones.

3.2.6 System

A system is a logical entity intended to provide easier control over a single physical installation.

For example – a site may have an outdoor unit that is connected to 2 indoor units. Depending on the system (manufacturer/model), it may be impossible to switch operation modes (cool/heat) for a specific indoor unit without also changing the other indoor unit. The system concept allows easier control of that.

A site may hold multiple systems.

3.2.7 User

A user represents a user of the system. A user identifies itself using an email address and a password.

Each user has a set of permissions defining what he can do.

3.3 Authentication

Authentication is done using a JWT (json web token) with a 10 day expiry - the user provides his email/password and receives an access token.

The access token should be sent with every API call in the ***x-access-token*** request header.

NOTE: In the near future we plan to support the **OAUTH2.0 *authorization code* and *implicit* flows**.

3.4 Response format

Standard HTTP response codes are used:

- 200 would mean the request was successful
  - A JSON object will be returned, containing 2 keys:

Success – true

Data – where applicable, will contain the requested/created data

- 4XX would mean a user error (parameters, not found, unauthorized etc.)
  - A JSON object will be returned, containing 2 keys:
    - Success – false
    - Message – error message string
- 5XX would mean an internal server error
  - A JSON object will be returned, containing 2 keys:
    - Success – false
    - Message – error message string

3.5 Throttling and security

The client (based on its IP address) is limited to TBD API requests per minute, and to a total of TBD requests per hour, to avoid malicious use. Requests exceeding these limits will be discarded.

Specifically the authenticate API is even more limited – TBD calls per minute, and a total of TBD requests per hour. Requests exceeding these limits will be discarded.

If a client sends more than TBD consecutive messages without the authentication header, or with a bad authentication token, it will be blocked for TBD.

3.6 Methods

The API documentation is available at https://app.swaggerhub.com/apis-docs/coolautomation/CoolPlatform/1.0

4. Websocket

4.1 Overview

The websocket interface is used to provide real-time updates to the client regarding the entities he controls. These updates include:

- Device connection status change
- Unit added/removed
- Unit status change (temperature change, mode change etc.)

The websocket interface is not designed to accept commands from the client. Use the REST API to do that.

The use of the websocket interface is not mandatory, as the REST API can be used to poll for status. However, it is recommended to use it because:

- Use of the polling REST API calls is limited to TBD requests per minute
- Some data needs to be discovered by the client (for example: unit added – client will need to notice that a new unit appeared instead of receiving an event)
- Some events have immediate value to the client (or example: device disconnect)

4.2 Connection establishment

A websocket connection is created by initiating a websocket connection to the TBD endpoint.

4.3 Authentication

The client will use the token acquired via the REST ***authenticate*** call to authenticate itself with the websocket server.

Once the client created the connection, the server expects to receive the authentication message within TBD seconds. If the message does not arrive within that timeframe, or if the first message that arrives is not the authentication message, the server closes the connection.

4.4 Throttling and security

The client (based on its IP address) is limited to TBD websocket creation attempts per minute, and to a total of TBD attempts per hour to avoid malicious use. Requests exceeding these limits will be discarded.

Client to server messages on this channel are limited to TBD per minute, and a total of TBD per hour. If this limit is exceeded the connection will be dropped by the server, and further connection attempts from this client will be blocked for TBD.

If any messages other than the expected types (authentication, keepalive) is received, the server will drop the connection.

The authentication message must be sent as the first message from the client, and it should be sent within TBD seconds of the connection establishment. Otherwise the server will drop the connection.

This message should be only sent once when the connection is established. If the client tries to send this message again, the server drops the connection.

4.5 Protocol

All messages on the websocket are JSON objects.

4.5.1 Authentication message

4.5.1.1 Direction

Client to server

4.5.1.2 Description

This message is intended to identify the client to the server.

4.5.1.3 Format

TBD

4.5.2 Keepalive message

4.5.2.1 Direction

Bidirectional

4.5.2.2 Description

This message is intended to validate on the application level that the connection is still alive.

The server will send this message once every TBD seconds. The client should respond with this message too.

If the server does not receive a reply by the next timeframe it is supposed to send a ping message, it will try again. If a reply to this attempt is not received by the time to send another keepalive message, the server drops the connection.

4.5.2.3 Format

TBD

4.5.3 Device connection status message

4.5.3.1 Direction

Server to client

4.5.3.2 Description

This message is intended to notify the client of a connection status change for a device the user operating the client has permissions to see.

4.5.3.3 Format

TBD

4.5.4 Unit added/removed to device message

4.5.4.1 Direction

Server to client

4.5.4.2 Description

This message is intended to notify the client that a unit was added or removed from a device the user operating the client has permissions to see.

4.5.4.3 Format

TBD

4.5.5 Unit status message

4.5.5.1 Direction

Server to client

4.5.5.2 Description

This message is intended to notify the client of a unit status change (temperature change, mode changed etc.) for a unit the user operating the client has permissions to see.

4.5.5.3 Format

TBD

4.5.6 Notification message

4.5.6.1 Direction

Server to client

4.5.6.2 Description

This message is intended to notify the client that a user defined alert was triggered or cleared (temperature threshold crossed, user connection etc.) for an entity the user operating the client has permissions to see.

4.5.6.3 Format

TBD

4.5.7 Extended stats (pro) message

4.5.7.1 Direction

Server to client

4.5.7.2 Description

This message is intended to provide extended stats (aka pro stats) for an entity the user operating the client has permissions to see.

The user must have an appropriate license to receive this message.

4.5.7.3 Format

TBD

5. Simple usage flow

This section details a basic usage flow of the platform.