TrainLink API

Release 0.1

Oct 31, 2020

Contents

CHAPTER 1

Welcome to TrainLink API's documentation!

The TrainLink API allows a DCC++ EX BaseStation to be accessed over a local network. In this documentation you will find the getting started guide along with reference for all the different API calls

1.1 Coming Soon!

This is currently a work in progress, with more to be added shortly.

CHAPTER 2

Getting Started

This page details how to download and install the API, as well as using the demo page.

2.1 Downloading TrainLink

You will need to download the files for TrainLink from the releases section of the GitHub repository. To do this you will need to go to the TrainLink repository and click on the latest release that isn't a prerelease, unless you specifically want to use features only in that release. Finally, download the correct file for your OS (.zip for Windows, others use tar.gz) then extract it to a safe place.

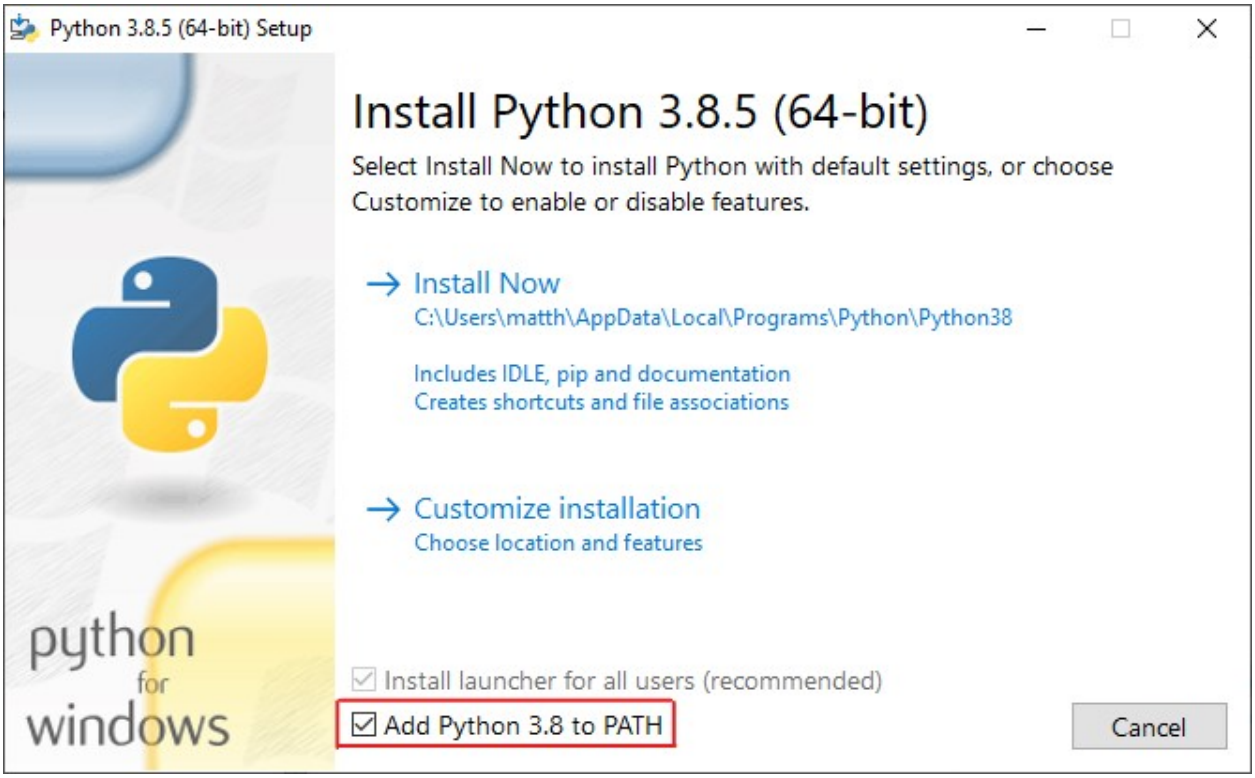
2.2 Setting up the server

First you will need to set up the server on the computer that the BaseStation will be plugged into. To do this you need to follow the steps below:

2.2.1 Install Python

To install Python, you need to head here then download the correct version for the operating system you are using. If you are using MacOS or Linux, you can skip this stage as python is preinstalled on your OS.

Note: If you are using Windows, be sure to tick the box labled 'Add Python to Path' as this will enable you to use properly.



2.2.2 Install dependencies

Then, you need to open up your terminal into the root of the files you downloaded earlier and run

```
$ pip install -r requirements.txt
```

This will install all of the required pip packages needed to run the server.

2.2.3 Run the server

Thats it! All the installing is done, and all thats left to do is run `server.py` in the API Server folder. However, you might want to check out the *Configuring the server* section on the *API Calls* page for details on how to customise the server.

CHAPTER 3

Using the API

This page details general usage of the API and how to implement it into your code.

3.1 Configuring the Server

The server is configured using the file called `config.xml`. Here is how to use this file:

3.1.1 Cabs

In this section, you can define a cab to add a phonetic name to use instead the DCC address set up in the decoder's CV. Cabs are defined in the format:

```
<cab>
    <name>name of train</name>
    <address>train address</address>
</cab>
```

You can delete the default `Train1` and `Train2` profiles, these are just included for testing and example purposes.

3.1.2 Server core

This is where the ports etc for the server are set up.

ip

Controls the IP addresses the server will listen to connections on

```
<!-- sets ip to 0.0.0.0 (all connections) -->
<ip>auto</ip>

<!-- sets ip to 127.0.0.1 (local only) -->
<ip>local</ip>
```

port

Controls the TCP/IP port the server listens for connections on

```
<!-- sets the port to 6789  -->
<port>auto</port>

<!-- sets the port to 1234 -->
<port>1234</port>
```

serialPort

Controls which serial port the server attempts to contact the DCC++ BaseStation on.

```
<!-- sets the port to COM1 -->
<serialPort>COM1</serialPort>
```

3.1.3 Debug

This controls whether the debug statements are enabled or not. If `enableDebug` is set to *True*, then debug statements will be showen in both the server output and also the console in the browser (accesible via F12 on most browsers, otherwise look for *developer tools*).

3.2 Using TrainLink for Websites

TrainLink for Websites uses javascript to communicate with the webserver. This gives the benefit of the page being able to constantly update, even after the webpage has loaded.

3.2.1 Adding the .js to your code

To enable access to all the TrainLink functions, you need to add one line of code to the top of your body in each HTML file where the functions are needed. The line of code is this:

```
<script src="path/to/trainlink.js"></script>
```

I recommend putting the `trainlink.js` file in a folder named something like *js*. In which case, your path to the .js would be `js/trainlink.js`. `trainlink.js` can be found in *API Libraries/TrainLink for Websites* in the file structure you downloaded from the TrainLink API GitHub page. If you haven't done this yet head to *Getting Started*.

3.2.2 Setting up your code for TrainLink

Now you have added the `trainlink.js` to your page, you can access all the API functions as long as you call `trainlink()`. If you want more information on how to do this, have a look at *API Calls*. There are a couple of compulsory functions that you need in your code at some point and these are `update()` and `config()`. They should be structured like this:

```
function config(data) {
    /* Code to be run when a connection is estabilised to the server */
}

function update(data) {
    /* Code to be run when the server sends an update packet */
}
```

Contents of the data variable

data.updateType will store the type of update that the packet is.

Update type	Data values available	Usage
"cab"	data.cab	Stores the address of the cab that the packet refers to
	data.speed	Stores the new speed of the cab
	data.direction	Stores the direction of the cab
"power"	data.state	The current state of the trackpower
"config"	data.cabs	A list of all the cabs defined in the xml
	data.debug	If debug is enabled in the server config.xml

Although a list of defined cabs is provided to the client when they connect, you can still address cabs not on the list. These will be added to the internal arrays when you first use each address.

CHAPTER 4

API Calls

This page details the calls available to programmers using the API on each platform.

4.1 Javascript Client

This is the library used for developing web applications. Due to the use of Javascript, pages written using this method can be run as files, rather than in a server.

`trainlink.`**`initateTrainLink`**(*ipAddress="127.0.0.1"*, *port="6789"*)

Parameters

- **`ipAddress`** – The local IP address of the server (127.0.0.1 restricts access to local machine only).
- **`port`** – The port of the server (6789 is the default).

Creates a link with a DCC++ BaseStation. If the server is running on your local machine, you can use `127.0.0.1`. Otherwise, use the IP address or hostname of the server. I recommend setting up a static IP for this machine or using it's hostname.

`trainlink.`**`setSpeed`**(*address*, *speed*, *direction=-1*)

Parameters

- **`address`** – The address of the cab that you want to change the speed of
- **`speed`** (*`int`*) – The new speed for the cab
- **`direction`** (*`int`*) – The direction for the cab. If the direction is set to -1, then the polarity of the speed value is used to set the direction. If the value of speed is negative, the cab will reverse. Likewise, if the speed is positive the cab will go forwards.

Sets the speed of the given cab. This function is very flexible as you can use either the direction argument or the polarity of the speed to change the direction. Also, for address, either the numerical decoder-set address can be used, or alternativly, the phonetic name set in the server's `config.xml`.

trainlink.**stopCab**(*address*)

Parameters **address** – The address of the cab that you want to stop

Stops the cab using the deceleration value set in the decoder. Like *setSpeed*, there are two ways of choosing the address. Refer to the setSpeed description for more details.

trainlink.**eStopCab**(*address*)

Parameters **address** – The address of the cab that you want to stop

Stops the cab **instantly**, ignoring the deceleration value set in the decoder. Like *setSpeed*, there are two ways of choosing the address. Refer to the setSpeed description for more details.

trainlink.**sendCommand**(*command*)

Parameters **command** – The command that will be sent to the BaseStation

Sends a command directly to the BaseStation. An example command would be `<t 1 3 126 1>`.

trainlink.**setPower**(*state*)

Parameters **state** – The power state the track will be set to (0 - off, 1 - on)

Changes whether power is applied to both the programming and main tracks. No power = no movement!

trainlink.**cabFunction**(*address*, *function*, *state=-1*)

Parameters

- **address** – The address of the cab you want to change the function of
- **function** (*int*) – The function number to change (e.g. 0)
- **state** (*int*) – The state to set the funtion to. 0 is off, 1 is on and -1 (the default state) toggles the function.

Changes a function for a cab. An example of this is lights and sounds (if a DCC sound decoder is fitted). Like the other functions, both the phonetic name and DCC address can be used for the address

CHAPTER 5

Internal workings of the API

A deeper dive on how the API works underneath the skin. Intended for advanced users only, use this page if you want to write your own application that works with TrainLink without using the API.

Coming Soon!

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

t

Index