WAXExpressTrade Documentation

Release 0.0.1

WAX ExpressTrade

Nov 27, 2020

General Documentation

This is a tutorial/documentation about the WAX ExpressTrade API.

CHAPTER 1

Issues

If you have troubles with the API, head over to the official GitHub repository.

Hint: The playlist for the videos can be found on YouTube

1.1 Getting Started

Hint: The video for this parts can be found here

This document will show you how to implement the WAX ExpressTrade to your website.

Info

If you already have a VGO API Key and an Ethereum Adress you can skip this section.

1.1.1 Basic information

The documentation requires a little bit of knowledge about the following points.

API

An application program interface (API) is a set of routines, protocols, and tools for building software applications. Basically, an API specifies how software components should interact. Additionally, APIs are used when programming graphical user interface (GUI) components. A good API makes it easier to develop a program by providing all the building blocks. A programmer then puts the blocks together. So this means it is even easier to implement the WAX ExpressTrade features.

Server & HTTP

So the server usually handles all the requests made to the API. For better structure you will need some kind of HTTP server to make requests to the WAX ExpressTrade API. HTTP related terms like GET and POST shouldn't be a problem as well.

JSON

JSON is very popular nowadays, especially with these so called REST-API's. The WAX ExpressTrade API uses JSON to format the information. Every requests needs some kind of JSON-structure. The response of every API-endpoint will also always return an JSON-Object (Unless you encounter some kind of other error).

1.1.2 Requirements

To get in touch with WAX ExpressTrade API it only requires three things. We will go over them, what they do and how to fulfill them.

VGO API Key

For most of the WAX ExpressTrade API you will need to provide an API Key. (To identify yourself) You can simply generate one by using the following method.

1. Open up some kind of console (Windows Command Prompt, Linux/Mac Terminal)
2. Copy the line beneath and change the `site_url` and `display_name` to a valid value. (This should be the name of your website)

```
$ curl -d '{"site_url":"http://yoursite.com","display_name":"yoursite"}' -H "Content-
↪Type: application/json" -X POST https://api-trade.opskins.com/IUser/CreateVCaseUser/
↪v1/
```

3. Wait for the response

Listing 1: You probably will receive a response like this (If not, the error should be returned. Try to adjust/edit your `curl` line and try again)

```
{
   "status":1,
   "time":1535545650,
   "response":{
      "api_key":"some kind of api key",
      "user":{
         "id":-868,
         "steam_id":"",
         "display_name":"yoursite",
         "avatar":null,
         "twofactor_enabled":false,
         "api_key_exists":true,
         "sms_phone":null,
         "contact_email":null,
         "allow_twofactor_code_reuse":false,
         "inventory_is_private":true,
         "auto_accept_gift_trades":false,
         "vcase_restricted":true
```

(continues on next page)

(continued from previous page)

```
        }
    }
}
```

4. Your "VGO API Key" will be placed in the JSON-Object with the name "api_key". Save it, you will need it later on

Ethereum Address

An Ethereum Address is needed for the key requests (to open cases). There are tons of Ethereum Wallets on the internet, nearly all of the are free to use. For the ease of use I'd recommend MetaMask or if you want it more secure maybe go for something like Coinbase. The choice is up to you. At some point you should have an Ethereum Adress looking something like this: `0xde0B295669a9FD93d5F28D9Ec85E40f4cb697BAe`

1.2 Initial Testing

This specific section should help you to verify and understand how and if you requests work.

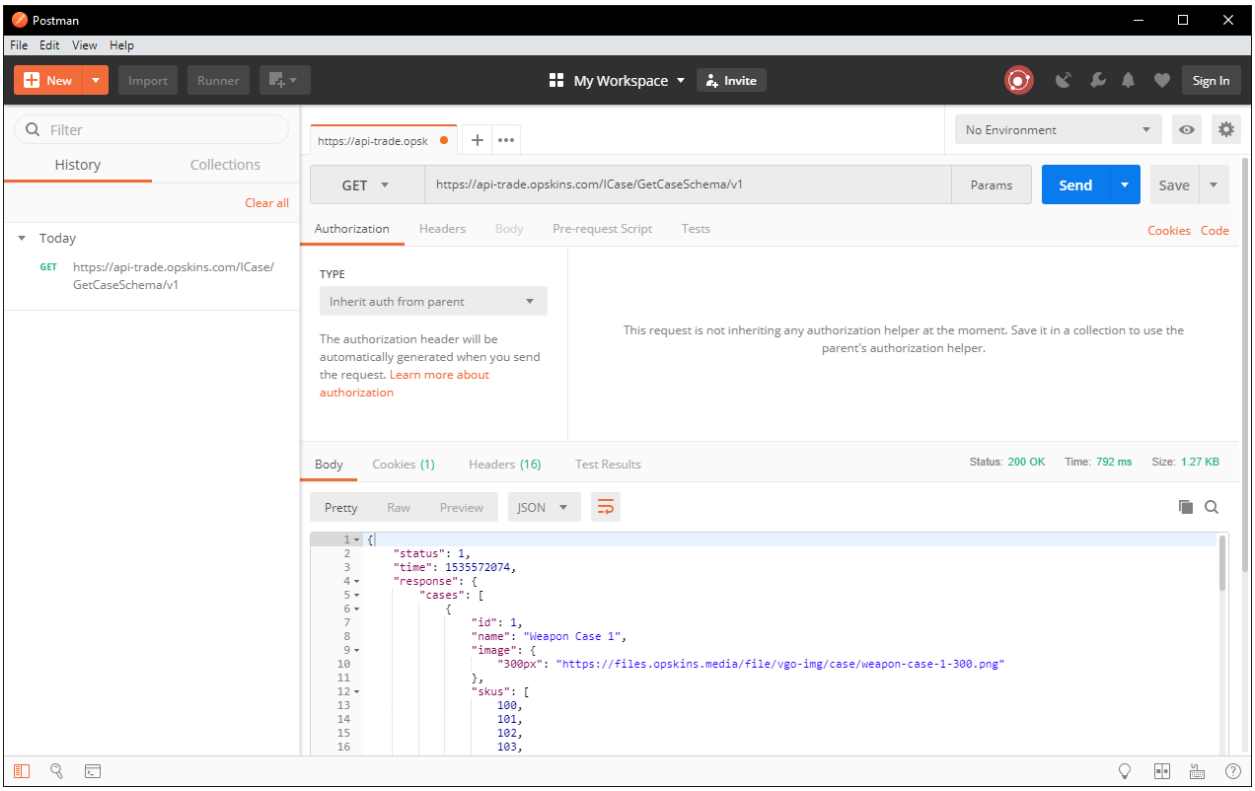
1.2.1 The First Request

To make your first request I recommend a program like Postman.

But why would we need that?
Simply because it is easier to use if you want to make requests to the API and you can quickly test and check specific API endpoints. You can add your API Key which we will need for some actions and you can format the JSON much more easier as well.

Once you downloaded the program you will be prompted to sign in (You can skip that at the bottom). Afterwards we can try to make our first request to the WAX ExpressTrade API.

Our first request will not require an VGO API Key. At the top input bar where it says `Enter request URL` type in something like `https://api-trade.opskins.com/ICase/GetCaseSchema/v1` and choose `GET` as your request method.

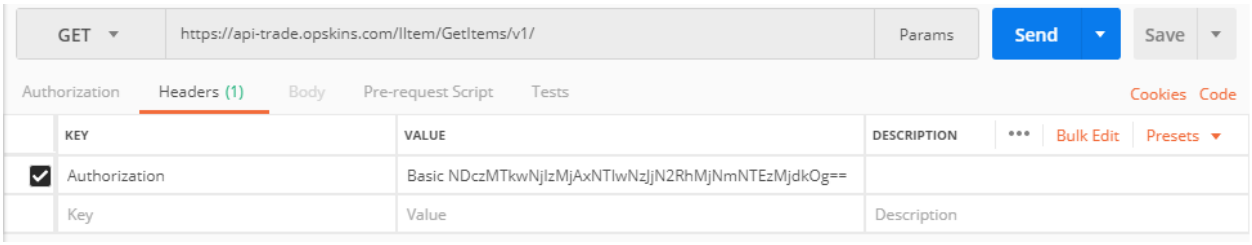


If it was successful, you should see something like that in the image above.
Well that was easy, but you also could have done that request in your browser by typing in the URL.

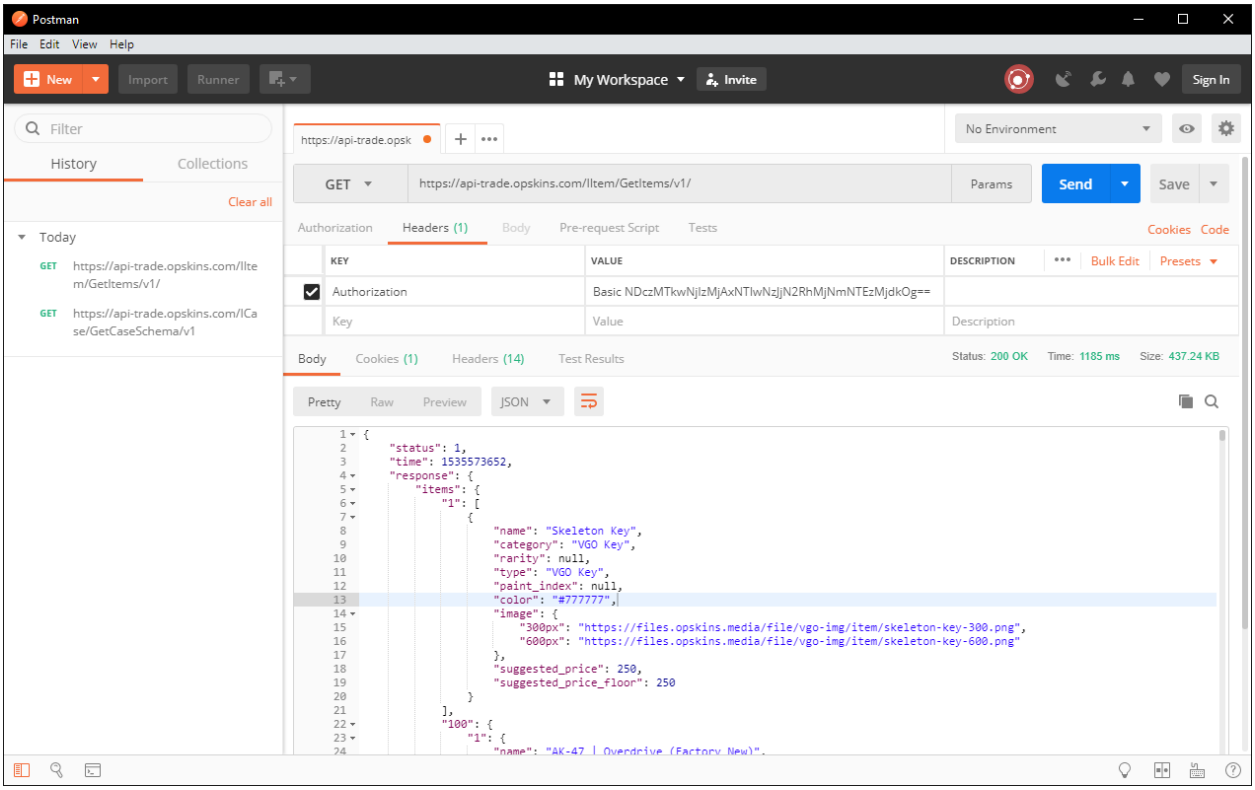
1.2.2 The Second Request

Now we want to make a request that requires the API Key. The WAX ExpressTrade API has a "Basic Authorization" System. This means we have to convert your VGO API Key to Base64 before we make our request.

1. Head over to https://www.base64encode.org/.
2. Type in your API Key and add a colon `:` at the end of it so it looks something like this `47319062320152072c7da23f51327d:`
3. Hit **ENCODE** and copy the Base64 string
4. Head over to Postman again and click on the tab "Headers"
5. For the **Key** you need to type in `Authorization` and for the **Value** you have to type in `Basic YOUR_BASE64_STRING`



6. Once this header is set you can choose an API endpoint that requires the API key. In this case I chose `https://api-trade.opskins.com/IItem/GetItems/v1/` to get all the items and stats to each item in the cases.



If you messed up something it probably will have a response like

```
{
  "status": 401,
  "time": 1535573754,
  "message": "API Key Required"
}
```

Troubleshooting

1. Check if you have ticked the box to send the `Authorization Header`
2. Check your `Key` and `Value` for this specific header
3. Check your Base64-String. Does it end with "==" or not?
4. Check your String before converting it to Base64. Have you added to colon `:`?
5. Try a new API Key, or this one, used for the tutorial `NDczMTkwNjIzMjAxNTIwNzJjN2RhMjNmNTEzMjdkOg==`

1.2.3 Conclusion

So you have successfully requested data from the WAX ExpressTrade API. You can also POST data to the API. Search for a POST request in the "Advanced Documentation", add your API Key like we did in the tutorial above and POST your data.

Of course you can try different endpoints, scroll through all the JSON-Data the server has returned and check it's content or use other request types such as POST.
But all in all this is pretty easy isn't it? But we want to implement this into a website, not just requesting data in Postman to view it.

1.3 Setup

We want to split up the implementation into 2 main parts

1.3.1 Backend (Server)

So the server (also called backend) should handle all our requests made to the WAX ExpressTrade API.
You may wonder why we should do that because we could make our requests through the frontend (Website) as well. There are a couple of reasons starting with

1. **You need some kind of server to host your website**
    - If you have a more static site, a simple Apache or NGINX server will probably do it
    - But this tutorial focuses more on developing a web app and less on a static website
    - Creating your own simple HTTP server will give you more flexibility
2. **The VGO API Key**
    - As mentioned in the previous section we need the VGO API Key for most of our requests. So if you want to make a request, for example to initiate a case opening, you would need the API Key of course.
      Keep in mind that you do not want to share you API Key with others, because it is linked to your website and website name.
      If you make your requests through the frontend (website) you have to save your API Key to the HTTP request which means it is public and everybody can abuse it.
      You need to make sure your server works as a middleware between the WAX ExpressTrade API and your frontend (website).

1.3.2 Frontend (Website)

So the frontend (website) will be completely up to you. How you design it, how your animations will work, how you will implement the requests.
Here are some recommendations you could use to make an advanced, progressive web app

1. **The most known and common framework React**
    - easy to start and learn
    - lot's of free tutorials on YouTube (Check out Traversy Media for excellent and detailed tutorials)
    - Direct links to the tutorial React JS Crash Course and a playlist about Learning React
2. **Another great frontend framework Angular**
    - Nearly same features as React but everything in TypeScript instead of JavaScript
    - Video Angular In 60 Minutes

3. Vue.js
4. Very simple and old school with plain JavaScript and jQuery (not recommended for progressive web apps)

1.4 Backend (Server)

Hint: The video for this parts can be found here

This section will be about how to request data from the WAX ExpressTrade API and how to bring this data to the frontend (website) so the user.
For this purpose we will create a simple HTTP server in JavaScript with the help of Node.js.

Hint: All the code written below is also available on GitHub.

1.4.1 Requirements

In my opinion the easiest way to get started with an HTTP server is to create an Node.js HTTP Server. We will use the very popular package express for this purpose.
But first let's talk about the requirements to get started

1. **We will need Node.js.**
    - Installation guide for Windows
    - Installation guide for Linux (all versions)
2. **A more advanced text editor than Notepad in Windows to write our code. Recommended free text editors:**
    - Atom
    - Visual Studio Code (only for Windows)
    - Sublime Text
3. **(Optional and only for Windows users) A different console**
    - I personally do not like the command prompt in Windows
    - My favorite console so far is the GIT Console (You will see why later on)

Once you installed Node.js and chose your text editor we can start coding. I will use the following setup for this tutorial

- Node version 8.11.4 (you can check it by simply opening a console and by typing in `node --version`)
- Atom (1.30.0) as my text editor with the File Icons Package (just for cosmetic)
- Because I am on Windows 10, I am going to use the GIT console

1.4.2 Step 1

We need to create all the dependencies for our Node.js Server

1. Create a folder where you want to put your server
2. Open up a console and direct into that newly created directory (if you are using the GIT console on Windows, you can simply right-click and choose "Git Bash Here")
3. **Once you are in the desired directory type in `npm init` to start the setup**
    - **package name** Choose the name of your project. Usually it picks up the name of your folder. I will call it `wax_tut`.
    - **version** Doesn't matter at all, just hit Enter
    - **description** Type in something like "This is a tutorial on how to implement the WAX ExpressTrade API" or if you want to skip it, hit Enter
    - **entry point** This is important. Here you have to choose the name of your main server file. You can either choose the default value "index.js" or rename it to something like `server.js`. I will call it `server.js`
    - **test command** Skip this for now
    - **git repository** If you already got an git repository, you can type it in here. Otherwise you can skip this.
    - **keywords** Define some or skip, doesn't really matter
    - **Author** Basically your name
    - **license** Just hit Enter
    - Check your inputs and confirm with `yes` (Don't worry you can change the values afterwards)
4. Create a JavaScript file with the name you chose at `entry point`. In this case you need to create a file called *server.js*
5. Open this file with your desired text editor

1.4.3 Step 2

In Step 1 we created our project and our server file. In this part we will set up the basic structure of our server and test it if everything works.

1. Double-Check your folder if you see both the `package.json` and `server.js` file.
2. **Install the dependencies we need for the HTTP Server**
    - As mentioned earlier, we will use express for our HTTP server.
    - Open up your console again and type in `npm install --save express`. This will install the express package and with the option `--save` you will also save the dependency to your `package.json` file, so you always know what dependencies are install in your "node_modules" folder
3. Paste this basic HTTP server into your `server.js` file

```
var express = require('express')
var app = express()

app.get('/', function (req, res) {
  res.send('Hello World')
```

(continues on next page)

(continued from previous page)

```
})

app.listen(3000)
```

Code explanation

- `var express = require('express')` First of all we implement our express package so we can use all it's functions.
- `var app = express()` Afterwards we create an instance of the class so we can access every function.
- **Next we create our first route that will send a response once it gets called.**
  - But the method has to be GET and the route must be "/" to access it.
- The last part is on which port our server should listen. Keep `3000` for now, but you can always change it as you wish.

Let's try it out

So if you pasted in this code and saved it to the `server.js` file, go over to your console again (make sure you are in the right directory) and type in `node server.js` to boot up the server.

Note: For testing purpose I recommend the package `nodemon` so the server simply restarts every time we change something. Otherwise you would have to cancel it with CTRL+C and restart it with `node server.js`. Install the package with `npm install -g nodemon` and start your server with `nodemon server.js`

Once your server is running either go to your browser or open up Postman and type in `127.0.0.1:3000`. (This is your local IP-Address on your computer)
If you are being greeted with "Hello World" you have successfully created your HTTP server in JavaScript

1.4.4 Step 3

Your HTTP server is up and running and you have successfully accessed it. Time to request data from the WAX WAX ExpressTrade API.
So the basic idea behind this

1. We request something from OUR HTTP server like we did to check if the server is working e.g. `127.0.0.1:3000/caseschema`.
2. The server should request data from the WAX ExpressTrade API before it sends the response back to us.
3. Instead of "Hello World" we should see data from the ExpressTrade API.

Let's see how we can do that.

1. We need another package to request data from other sources. For this we will use axios.
2. Head to your console and press CTRL+C to stop the server.
3. Type in `npm install --save axios` to install the package.

4. Head over to the text editor and change the code.

```
// Packages
var express = require('express')
var Axios = require('axios');

// Variables
var vgoURL = "https://api-trade.opskins.com";
var vgoAPIKey = "YOUR API KEY";

// Variable for WAX ExpressTrade requests
var vgoAPI = Axios.create({
  baseURL: vgoURL,
  headers: {
    Authorization: 'Basic ' + Buffer.from(vgoAPIKey + ':').toString('base64'),
  },
});

// create express server
var app = express()

// route 1 (simple GET request)
app.get('/caseschema', async function (req, res) {
  let response = await vgoAPI.get('/ICase/GetCaseSchema/v1');
  res.send(response.data);
});

// start server
var listener = app.listen(3000, function() {
  console.log('Listening on port ' + listener.address().port);
});
```

So the code changed quite a bit, let's go over it real quick and talk about how you can customize it.
So following changes have been made:

1. I added the axios package that we use to request the data from the API
2. Two new variables have been added `vgoURL` and `vgoAPIKey`. You do not need to touch the `vgoURL` but you have to paste in your generated VGO API Key (**Not the Base64 encoded one, but the plain VGP API Key**)
3. Afterwards I created a new instance of axios and saved it to the variable `vgoAPI`. This comes in pretty handy because later on we only need to call the variable `vgoAPI` to make a request to the actual API. This request will always have the "Authorization Key" in it, so we don't have to worry about that.
4. **I made some changes to the `GET /` route and renamed it `GET /caseschema`. It now requests data from the WAX Expr**

    - The quick and easy answer is, that the `async` which stands for "asynchronous" prevents Node.js to do things at the same time.
    - "Asynchronous" means it is not existing or occurring at the same time.
    - If don't use the async/await your response would be empty because Node.js is trying to do this one function at the same time.
    - The request/response to the WAX ExpressTrade API needs more time than our server needs to respond to the `GET /caseschema` call.

5. No we send the `response.data` as a response instead of "Hello World"
6. I changed to code on how the server starts, it does the same but will show you a console entry when the server has been started successfully

So once again either head over to your browser or Postman (this time I highly recommend Postman because it will be a lot of JSON data) and try out what results you get.
You should get a list off all the available cases (name, image) and what item skus are in it.

1.4.5 Additional Examples

This section is about some another examples of how to use the WAX ExpressTrade API.

Example 1

Listing 2: GET request with additional parameters

```
// route 2 (GET request with additional parameters)
app.get('/keycount', async function (req, res) {

  var trade_url = req.query.trade_url;

  if (!trade_url) {
    return res.send("No trade url provided");
  }

  let response = await vgoAPI.get(`/ICaseSite/GetKeyCount/v1?trade_url=${trade_url}`);
  res.send(response.data);
});
```

Explanation

- So this this route requests the key amount a specific user has. Again we use a GET method but this time we have to provide an additional parameter in the URL called `trade_url`.
- In express you can access those URL parameters via `req.query.YOUR_PARAMETER`. Easy to use and fast to understand. So you save the `req.query.trade_url` parameter to a variable called `trade_url`.
- Afterwards we quickly check for this parameter if it has been provided. If not provided we simply return a response. (Sidenote: You need to type `return res.send("...")` to stop the function from continuing. If there is no `return` provided, the code will continue and fail at the request.)
- Anyways, with a provided `trade_url` we request the data from the server. Take a look and remember on how to add the query to the URL `?trade_url=${trade_url}`, you will probably need that for other requests as well.
- And for the last part we send the response back to the request.

Note: You can only request the keycount of people that have been registered on OPSkins.

How To Test It

Note: If you are familiar on URL parameters and how to use them properly you can probably skip this part and just test your newly created route.

For those that are not comfortable with, I got you

- URL parameters are additional string in the URL.
- For example if you go to `https://www.google.com` you won't see any parameters in the URL.
- But once you search for something like *wax expresstrade*, Google will add parameters to its URL. The URL will look something like this `https://www.google.com/search?source=hp&ei=5m6IW6r0LY_YwQL2ma_ACg ...` (Not the full URL)
- So the URL has a parameter for `source` and `ei` and so on and so forth.
- In our case the parameter should have the name `trade_url` and has to be provided.
- This means our URL has to look like this `http://127.0.0.1:3000/keycount?trade_url=VALID_TRADE_URL` to successfully request data from the API

Note: If you encounter some kind of error it will probably be something like `UnhandledPromiseRejectionWarning` and the API responded with something like `Request failed with status code 400`. I will talk about that in the next example **(Explanation, 4.)**.

Example 2

Listing 3: POST request with additional parameters

```
// used to access the body as it was JSON
app.use(express.json())

// route 3 (POST request with additional parameters)
var affiliateAddress = "YOUR ETHEREUM ADRESS";

app.post('/opencase', function (req, res) {

  if (!req.body.trade_url  || !req.body.caseId || !req.body.amount) {
    return res.send("One or more parameters are missing!");
  }

  vgoAPI.post('/ICaseSite/SendKeyRequest/v1', {
    trade_url : req.body.trade_url,
    case_id: req.body.caseId,
    amount: req.body.amount,
    affiliate_eth_address: affiliateAddress
  })
  .then(function(response) {
    res.send(response.data);
  })
  .catch(function(error) {
    res.send(error.response.data.message);
  })
```

(continues on next page)

(continued from previous page)

```
});
```

Explanation

So first of all this is a more advanced example but a very efficient and easy to understand one.

1. We are using a middleware for express to access the body as it was JSON. `app.use(express.json())`
2. We created a variable `affiliateAddress`. You need to put your Ethereum Address to receive the commission for an opened case.
3. A new route has been created, but this time we do not use the GET method, for this example we are using the POST method.
4. Once again we check for three parameters that must be included in the request. But this time we are sending it via the body so the parameters won't be in the URL.
5. **A new way of requesting data from the WAX ExpressTrade API**
    - First of all we are using the POST method again, that is very important because only the POST method will be recognized on this endpoint
    - Second of all we are using a better way of handling the response and sending it back. This means we are using the option of "Promises". Thankfully axios got our back and provides a great and standard way of handling "Promises"
    - So we do not save the response to a variable called `response` like we did in all the other routes. Instead we use the `then()` callback option.
    - Once our request has succeeded, the `then()` function will be called and we can simply send our response as we did in the previous routes
    - The big advantage of "Promises" is the `catch()` function.
    - If there is any error it will be handled by this function. If the request wasn't successful, the "message" can be found by the given query `error.response.data.message` (the data is an JSON-Object)

Caution: If you are thinking of serious development I highly recommend using "Promises". Not using "Promises" actually is deprecate!

1.5 Frontend (Website)

Hint: The video for this parts can be found here

This specific section is all about on how to use your "middleware API" for your purpose. If you already know how to build a website and make request, handle your data or errors you can skip this section and head over to the **Advanced Documentation** to see the WAX ExpressTrade API documentation.

In this section I want to show you how you can implement your "middleware API" into an Angular web app.

Caution: DISCLAIMER This section is just a basic tutorial on how to get started with implementing the API on a website. If you know how to do it on your own just skip this!

Hint: All the code written below is also available on GitHub.

1.5.1 Angular

Angular definitely more complex than other frontend frameworks, this is why I want to give you a quick overview.

Requirements

1. **We will need Node.js.**
    - Installation guide for Windows
    - Installation guide for Linux (all versions)
2. **A more advanced text editor than Notepad in Windows to write our code. Recommended free text editors:**
    - Atom
    - Visual Studio Code
    - Sublime Text
3. **(Optional and only for Windows users) A different console**
    - I personally do not like the command prompt in Windows
    - My favorite console so far is the GIT Console (You will see why later on)
4. **The Angular CLI**
    - You can simply download Angular CLI by typing `npm install -g @angular/cli` into your console
    - We need that to create and build our Angular application

Once you installed Node.js and chose your text editor we can start coding. I will use the following setup for this tutorial

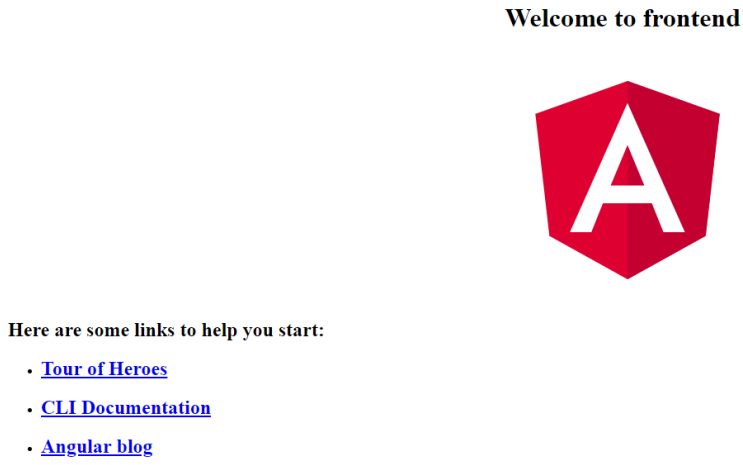
- Node version 8.11.4 (you can check it by simply opening a console and by typing in `node --version`)
- Atom (1.30.0) as my text editor with the File Icons Package (just for cosmetic)
- Because I am on Windows 10, I am going to use the GIT console

1.5.2 Step 1

Creating an setting up our application.

1. **To create a new application type `ng new PROJECT-NAME` in a newly opened console (watch out for the path in the conso**
    - You can choose any project name you like, as long as it matches the criteria
2. `cd` into your newly created project folder with `cd PROJECT-NAME`

3. Start your Angular application with `ng serve`
4. Open up your browser and type in `127.0.0.1:4200` or `localhost:4200`
5. You will be greeted with the default starter template



Caution: Make sure your API server is still up and running

1.5.3 Step 2

Time to add components, services and directories we need for our project.
If you have never worked with Angular, I recommend reading this article before you start coding your application.

1. Cancel the `ng serve` with CTRL+C
2. Change directory by typing `cd src/app/`
3. Create a new folder named **components** `mkdir components`
4. Change directory once again into the newly created directory `cd components/`
5. **Create your components with `ng g component COMPONENT-NAME`**
    - For this tutorial we will create a component named **navbar** and **main**

Now repeat these steps with services

1. We have to change directory to `src/app/` again because now we are in `src/app/componets`. Simply type `cd ..`
2. Create a new folder named **services** `mkdir services` and `cd` into it
3. **Create your services by typing `ng g service SERVICE-NAME`**
    - For this tutorial we will create a service named `api`

That's it for creating directories, components and files (You could generate all these things in the `src/app/` directory but it will become very messy.)

You can now serve your application once again with `ng serve`

1.5.4 Step 3

Open up your text editor and add your Angular application folder as a project folder.

1. Search for the file `app.module.ts` in `src/app/`.
2. Your components should already be added to this file (If not, just look at the code given below). Your code should look something like this

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';
import { NavbarComponent } from './components/navbar/navbar.component';
import { MainComponent } from './components/main/main.component';

@NgModule({
  declarations: [
    AppComponent,
    NavbarComponent,
    MainComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

3. Add your "API Service" by typing `import { ApiService } from './services/api.service';` and add the `ApiService Class` to the providers.

```
import { ApiService } from './services/api.service';

providers: [ApiService],
```

4. **Next we have to add the HttpClientModule to our `app.module.ts` file to make HTTP requests to the server.**

    - Simply import it at the top by typing `import { HttpClientModule } from '@angular/common/http';`
    - And add the `HttpClientModule` to the imports at the bottom

5. **(optional) The last thing we are going to add is Bootstrap to have some kind of nice design and a better UI**

    - Head over to https://getbootstrap.com/ and download the latest version (currently 4.1.3)
    - Go for the **Compiled CSS and JS** Bootstrap version
    - **Go to https://jquery.com/download/ and download the latest version of jQuery (go for the "compresed" one)**

  - You are better off if you hover of the "Download the compressed, production jQuery 3.3.1" link and right-click on it to "Save link as"
  - Save it to the Desktop for now

- Head to your Angular project folder and search for the `src/assets/` directory via your explorer
- Unzip the Bootstrap archive and copy the `css` and `js` folder into the `src/assets` directory
- Copy the jQuery.js file you saved on the Desktop into the `src/assets/js` folder
- Open up your text editor and search for the `angular.json` file in the project folder
- Search for line 25, you should see an entry called `styles`. Copy this path above the `"src/styles.css"` one `"src/assets/css/bootstrap.min.css"`
- Underneath the `style` entry there also should be `scripts` - Add the two following lines - `"src/assets/js/jquery-3.3.1.min.js"` - `"src/assets/js/bootstrap.min.js"` - Don't mess up the order of this two line and don't forget a comma after the jquery entry

It should look like this afterwards

```
{
  "styles": [
    "src/assets/css/bootstrap.min.css",
    "src/styles.css"
  ],
  "scripts": [
    "src/assets/js/jquery-3.3.1.min.js",
    "src/assets/js/bootstrap.min.js"
  ]
}
```

The last step is to restart your Angular application by canceling the serve with CTRL+C and `ng serve`.
That was it, we can now start coding our application.

1.5.5 Step 4

This section is about using our two created components and how to add some content on the page.

Implementing our components

Change the code in the `app.component.html` file to this

```
<app-navbar></app-navbar>
<app-main></app-main>
```

Explanation

With this two tags (which are Angular specific) we include our two created components `navbar` and `main`. The name `app-navbar` and `app-main` is specified by the component.
You can head over to your browser and check what has changed. You should see "navbar works" and "main works". If not, open up the browser developer console and check for any errors.

Adding content to the navbar and main component

1. Head over to the navbar component and search for the `navbar.component.html` file.
2. Copy the following template into the HTML file

Listing 4: Default navbar template found on https://getbootstrap.com/docs/4.1/components/navbar/

```
<nav class="navbar navbar-expand-lg navbar-light bg-light" style="margin-bottom: 1em;
↪">
  <div class="container">

    <a class="navbar-brand" href="#">WAX Tutorial</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="
↪#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false
↪" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav mr-auto">
        <li class="nav-item active">
          <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="https://trade.opskins.com/" target="_blank">Trade
↪</a>
        </li>
      </ul>
    </div>

  </div>
</nav>
```

I have changed the navbar a little bit so we can work with it even better.

Now let's change the main component

1. Search for the main component directory to edit the `main.component.html` file.
2. Add the following code

Listing 5: Default card component

```
<div class="container">
  <div class="row">
    <div class="col">
      <div class="card" style="width: 18rem;">
        <img class="card-img-top" src="https://via.placeholder.com/350x150" alt="Card␣
↪image cap">
        <div class="card-body">
          <h5 class="card-title">Card title</h5>
          <p class="card-text">Some quick example text to build on the card title and␣
↪make up the bulk of the card's content.</p>
          <a href="#" class="btn btn-primary">Go somewhere</a>
```

(continues on next page)

(continued from previous page)

```
        </div>
      </div>
    </div>
  </div>
</div>
```

Now save everything and head over to your browser to see the changes.

1.5.6 Step 5

Time to request data from the server and show it on our website.

Focusing on the API service

1. Open up the **api.service.ts** file
2. Copy in the following code, and check the "Explanation" below to understand what is going on.

```
import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

@Injectable({
  providedIn: 'root'
})
export class ApiService {

  constructor(private _http: HttpClient) { }

  getCaseSchema() {
    return this._http.get<CaseSchema>("http://127.0.0.1:3000/caseschema");
  }

}
```

(continues on next page)

(continued from previous page)

```
interface CaseSchema {
  status: any,
  time: any,
  response: any
}
```

Explanation

1. Once again we need the `HttpClient` to access HTTP requests
2. To use it, we need to define it in the constructor as well
3. We created a function that requests the "CaseSchema" from our API
4. This function can now be called from everywhere

Implementing The Service

So the next thing is to use our service function on the main component.

1. Open up the `main.component.ts` file (This file contains all the logic behind your static website)
2. Copy the code below and paste it into the file
3. Head to the explanation section to understand the code

```
import { Component, OnInit } from '@angular/core';

import { ApiService } from '../../services/api.service'

@Component({
  selector: 'app-main',
  templateUrl: './main.component.html',
  styleUrls: ['./main.component.css']
})
export class MainComponent implements OnInit {

  constructor(private _api: ApiService) {
    this._api.getCaseSchema()
      .subscribe(data => {
        console.log(data);
      }, error => {
        console.error(error);
      });
  }

  ngOnInit() { }

}
```

Explanation

1. So first of all we have to include our API service file into this one to use our function
2. Once again we create an instance of the class as seen in the `constructor()`

3. We call the function in the constructor so once the website is loading we are requesting the data immediately
4. **How does the function even work?**
    - `this._api.getCaseSchema()` this is just the way to call the function
    - `.subscribe()` Angular works with Observables which means you can only subscribe or unsubscribe to them (More about Observables can be found here)
    - Inside the `subscribe()` Method we can access two variables. One that contains our data called `data` in this case (You can call it whatever you want) and another one that contains any `errors` called error here (Again you can call it whatever you want)
    - It is recommended to use a better way of handling errors than simply log them to the console.
5. For now we are just printing the data to the browser developer console.

So let's head over to the browser and open up the console (Shortcut: F12)

Error

So you probably see an error saying `No 'Access-Control-Allow-Origin' header is present on the requested resource.`. This is more or less our "mistake" but actually this is a safety feature provided by your browser (I won't go any deeper on that topic).

We can fix this by creating a proxy for this request.

1. Create a file named `proxy.conf.json` in the root folder of your Angular project folder
2. Copy and paste the following code

```
{
  "/caseschema": {
    "target": "http://localhost:3000",
    "secure": false,
    "logLevel": "debug"
  }
}
```

3. **Edit the `angular.json` file.**
    - Head to line ~54 (this may differ, but just look for the entry `"serve"`)
    - Copy `"proxyConfig": "proxy.conf.json"` into the options so it looks like this

```
{
  "serve": {
    "builder": "@angular-devkit/build-angular:dev-server",
    "options": {
      "browserTarget": "frontend:build",
      "proxyConfig": "proxy.conf.json"
    },
    "configurations": {
      "production": {
        "browserTarget": "frontend:build:production"
      }
    }
  }
}
```

4. Restart the Angular serve instance by opening up the console, pressing CTRL+C and entering `ng serve` once again.
5. Adjust the URL in the `api.service.ts` file from `http://127.0.0.1:3000/caseschema` to just `/caseschema`

Now head to your browsers (maybe you have to refresh the website) and take a look in the console. You now should see the data from the WAX ExpressTrade API.

1.5.7 Step 6

Showing the data on the website.

For now we are only logging the data in the console but we actually want to show off the requested data. For that we have to adjust our code just a little bit.

1. **Changes in the `main.component.ts` file**
    - Create a variable above the `constructor()` like this `cases:  any = [];`
    - Change the line `console.log(data)` to `this.cases = data.response.cases;`
2. Changes in the `main.component.html` file

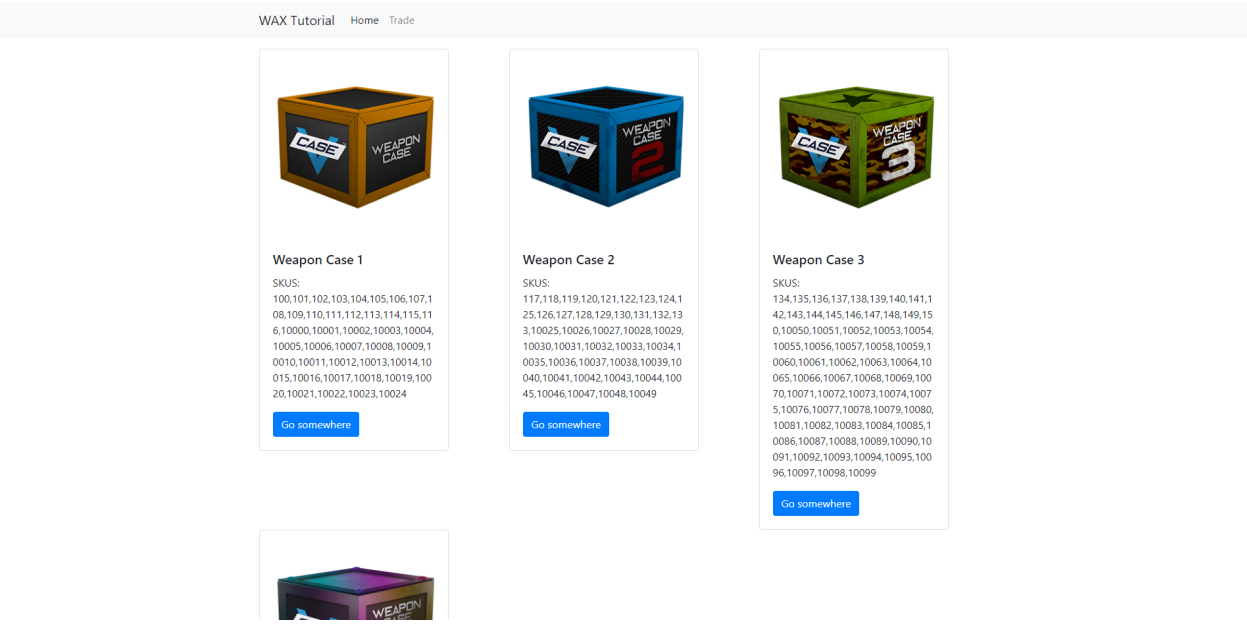
```
<div class="container">
  <div class="row">
    <div class="col" *ngFor="let case of cases">
      <div class="card" style="width: 18rem;">
        <img class="card-img-top" src="{{ case.image['300px'] }}" alt="Card image cap
↪">
        <div class="card-body">
          <h5 class="card-title">{{ case.name }}</h5>
          <p class="card-text">SKUS: <br> {{ case.skus}} </p>
          <a href="#" class="btn btn-primary">Go somewhere</a>
        </div>
      </div>
    </div>
  </div>
</div>
```

Explanation

1. **Changes to the `main.component.ts` file**
    - We created the variable so we can access it later on in the HTML file.
    - The only important thing is that the variable has to be an Array, so we can loop through it later on.
    - Because we do not need the data to be logged in the console we just replaced it.
    - Lastly we save the `data.response.cases` data to our created variable called `cases`
2. **Changes made to the `main.component.html` file**
    - The most important line probably is line three
    - As you can see we use the built in template syntax `*ngFor`. This is just a loop that creates as many objects as items stored in the array. In this case we have four array entries, so this loop creates four new object for us.

- This is a very convenient and easy way to create this objects. Especially because we can access each variable of the array individually. This means we simply use another template syntax to access data like “image”, “name” or “skus”
- To sum things up, this probably is the easiest and most convenient way to create an object for each, in this case, case.

That’s it, you can check the result in your browser. It should look like this



1.5.8 Additional Examples

If you are still reading this, I got more of some nice and easy examples on how to use the API.

This additional example will be about opening a specific case and handling the error if one occurs.

Example 1

This piece of code will allow a user to initiate a case opening.

Listing 6: `api.service.ts`

```
openCase(trade_url, caseId, amount) {
  return this._http.post<CaseSchema>('/opencase', { trade_url: trade_url, caseId:␣
↪caseId, amount: amount });
}
```

Listing 7: `main.component.ts`

```
openCase(caseId, amount) {

  // Usually you want to access the trade url via the request on your server
  let trade_url = "VALID TRADE URL";
```

(continues on next page)

(continued from previous page)

```
  if (!caseId || !amount)
    return alert("CaseId or amount not valid!");

  if (amount <= 0)
    return alert("You need to open at least 1 case!");


  this._api.openCase(trade_url, caseId, amount)
    .subscribe(data => {
      // this should be changed to some kind of alert as well, but I am unable to␣
↪test this properly because of insufficient keys
      console.log(data.response);
    }, error => {
      // simple browser alert with the error
      alert(error.error.text);
    });
}
```

Listing 8: `main.component.html`

```
<div class="container">
  <div class="row">
    <div class="col" *ngFor="let case of cases">
      <div class="card" style="width: 18rem;">
        <img class="card-img-top" src="{{ case.image['300px'] }}" alt="Card image cap
↪">
        <div class="card-body">
          <h5 class="card-title">{{ case.name }}</h5>
          <p class="card-text">SKUS: <br> {{ case.skus}}</p>
          <div class="input-group">
             <input type="number" class="form-control" placeholder="Keys" aria-label=
↪"keys" value="1" min="1" #keys>
             <div class="input-group-append">
               <span class="input-group-text btn btn-primary" (click)="openCase(case.
↪id, keys.value)">Open</span>
             </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

1.6 Quick Overview

1.6.1 API Interfaces

- *ICase*
- *ICaseSite*
- *IEthereum*
- *IItem*

- *ITrade*
- *IUser*

1.6.2 OAuth

OPSkins OAuth works automatically with WAX Trade. You can use OAuth to log users into your website via OPSkins and (if desired) perform actions on their behalf via the API. Please see OPSkins OAuth Docs for more information.

1.6.3 API Response

Direct URL to API: https://api-trade.opskins.com

All *successful* API responses have return data within the "response" object. A typical response may look like this:

```
{
    "status": 1,
    "time": 1528334546,
    "response": {
        "offer": {
            "some": "data"
        }
    }
}
```

If a response is paginated, the pagination details (*current_page* and *total_pages*) occur at the top-level of the object, not inside the *response* body.

1.6.4 Response Status Codes

All *status* codes and their titles can be found in the table below. In some instances, the *status* code may be an HTTP status code (e.g. 404). We recognize that mixing of these codes is not ideal and will fix this in the near future.

Status	Code
OK	1
GENERIC_USER_ACCOUNT_ERROR	102
ACCESS_DENIED	106
NOT_LOGGED_IN	108
NEEDS_TWOFACTOR	112
TWOFACTOR_INCORRECT	122
USERNAME_TAKEN	124
UNACCEPTABLE_USERNAME	126
GENERIC_INTERNAL_ERROR	202
DATABASE_ERROR	204
NOT_FOUND	206
BAD_STATE	208
NO_MATCHING_ITEMS_FOUND	210
CANNOT_CREATE_DIRECTORY	216
FILE_UPLOAD_ERROR	218
FILE_UPLOAD_ALREADY_EXISTS	220
CANNOT_DELETE_FILE	222

Continued on next page

Table 1 – continued from previous page

Status	Code
ALREADY_IN_THAT_STATE	226
LOCKED	228
DISABLED	234
MALFORMED_RESPONSE	236
EXPIRED	238
EMPTY_DATA	240
ITEM_NEEDS_REPAIR	246
ITEM_NOT_IN_INVENTORY	248
BAD_INPUT	302
UNACCEPTABLE_ITEM	304
DUPLICATE_ITEM	306
BAD_REQUEST	312
CAPTCHA_INVALID	316
RATE_LIMIT_EXCEEDED	318
MISSING_DEPENDENCY	326
REQUEST_OR_FILE_TOO_LARGE	330
UNACCEPTABLE_FILE_TYPE	332
THIRD_PARTY_UNAVAILABLE	408

1.6.5 Additional Notes

- On some endpoints you may be required to send a *twofactor_code*. Please see this comment if you need help.
- For transferring items from OPSkins to WAX ExpressTrade, see: OPSkins Docs: IInventory/TransferToTradeSite/v1

1.6.6 Dynamic Images

- On some items, you may see image URLs like so: `https://static.wax.io/d-img/...7cea75.png`
- Default image dimensions will be `300x300` (Width x Height) or lower (depending on the original image).
- You may request a different dimension by changing the end of the URL: `/600x600`, `/900x900`, etc.
- Best fit will be chosen automatically, so you may not always get the exact dimensions you choose.
- You can request the original (highest resolution) image with `/original`
- e.g. https://static.wax.io/d-img/dynamic-apps/img/cdff6f51e89199e8c9772535a17cea75.png/50x50
- e.g. https://static.wax.io/d-img/dynamic-apps/img/cdff6f51e89199e8c9772535a17cea75.png/600x600
- e.g. https://static.wax.io/d-img/dynamic-apps/img/cdff6f51e89199e8c9772535a17cea75.png/original

1.7 ICase

Endpoints to handle cases

Contents

1.7.1 ICase/GetCaseSchema

`GET https://api-trade.opskins.com/ICase/GetCaseSchema/v1`

Returns an object with all currently available cases.

Authentication

No auth required.

Input

Parameter	Type	Required	Description
cases	int-csv		A comma-separated list of case ids. If sent, output is limited to these specific cases.

Output

Parameter	Type	Description
cases	array-object	cases list
–id	int	Case ID
–name	string	Case name
–image	object	Case image URLS (Note: these images may change when remaining_opens hits 0)
—-300px	string	URL to 300px image
—-600px	string	URL to 600px image
—-900px	string	URL to 900px image
—-1800px	string	URL to 1800px image
—-2500px	string	URL to 2500px image
–skus	array	An array of item SKU in the case. Note that these may be updated overtime for vIRL cases & will not always be the same.
–key_amount_per_use	int	Number of keys required per 1 case opening
–max_opens	int	How many total items can be created from this case
–remaining_opens	int	How many items are remaining to be unboxed from this case. If this is 0, the case is depleted and cannot be opened anymore.

1.7.2 ICase/GetCaseOdds

`GET https://api-trade.opskins.com/ICase/GetCaseOdds/v1`

Authentication

No auth required.

Input

Parameter	Type	Required	Description
cases	int-csv		A comma-separated list of case ids. If sent, output is limited to these specific cases.

Output

Parameter	Type	Description
cases	array-object	An array of objects containing each case
–id	int	Case ID
–name	string	Case Name
–total_weight	string	Total weight of case in kilograms
–total_percent	string	Total percent, relative_percent's added together.
–odds	array-object	An array containing object lists of odds per sku
—-sku	int	Item sku for VGO, def_id for other apps/items
—-weight	string	Weight correponding to total_weight
—-relative_weight	string	Weight relative to all other items in the case
—-relative_percent	string	% chance of receiving this item, can be displayed to user. (relative_weight * 100)

Listing 9: Example Output (https://api-trade.opskins.com/ICase/GetCaseOdds/v1?cases=1)


```
{
"status": 1,
"time": 1545432800,
"response": {
    "cases": [
    {
        "id": 1,
        "name": "Weapon Case 1",
        "total_weight": "258917554",
        "total_percent": "99.99999999999986",
        "odds": [
        {
            "sku": 100,
            "weight": "780000",
            "relative_weight": "0.00301254197697",
            "relative_percent": "0.30125419769723"
        },
        {
            "sku": 101,
            "weight": "780000",
            "relative_weight": "0.00301254197697",
            "relative_percent": "0.30125419769723"
        },
        {
            "sku": 102,
            "weight": "10385844",
            "relative_weight": "0.04011255258498",
            "relative_percent": "4.01125525849823"
        },
        {
            "sku": 103,
            "weight": "10385844",
            "relative_weight": "0.04011255258498",
            "relative_percent": "4.01125525849823"
```


(continues on next page)

(continued from previous page)


```
            },
            {
                "sku": 104,
                "weight": "10385844",
                "relative_weight": "0.04011255258498",
                "relative_percent": "4.01125525849823"
            },
            {
                "sku": 105,
                "weight": "12177681",
                "relative_weight": "0.04703304512138",
                "relative_percent": "4.70330451213825"
            },
            {
                "sku": 106,
                "weight": "12177681",
                "relative_weight": "0.04703304512138",
                "relative_percent": "4.70330451213825"
            },
            {
                "sku": 107,
                "weight": "12177681",
                "relative_weight": "0.04703304512138",
                "relative_percent": "4.70330451213825"
            },
            {
                "sku": 108,
                "weight": "12177681",
                "relative_weight": "0.04703304512138",
                "relative_percent": "4.70330451213825"
            },
            {
                "sku": 109,
                "weight": "12177681",
                "relative_weight": "0.04703304512138",
                "relative_percent": "4.70330451213825"
            },
            {
                "sku": 110,
                "weight": "23580231",
                "relative_weight": "0.09107235347975",
                "relative_percent": "9.10723534797490"
            },
            {
                "sku": 111,
                "weight": "23580231",
                "relative_weight": "0.09107235347975",
                "relative_percent": "9.10723534797490"
            },
            {
                "sku": 112,
                "weight": "23580231",
                "relative_weight": "0.09107235347975",
                "relative_percent": "9.10723534797490"
            },
            {
                "sku": 113,
```


(continues on next page)

(continued from previous page)


```
            "weight": "23580231",
            "relative_weight": "0.09107235347975",
            "relative_percent": "9.10723534797490"
        },
        {
            "sku": 114,
            "weight": "23580231",
            "relative_weight": "0.09107235347975",
            "relative_percent": "9.10723534797490"
        },
        {
            "sku": 115,
            "weight": "23580231",
            "relative_weight": "0.09107235347975",
            "relative_percent": "9.10723534797490"
        },
        {
            "sku": 116,
            "weight": "23580231",
            "relative_weight": "0.09107235347975",
            "relative_percent": "9.10723534797490"
        },
        {
            "sku": 10000,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
        },
        {
            "sku": 10001,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
        },
        {
            "sku": 10002,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
        },
        {
            "sku": 10003,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
        },
        {
            "sku": 10004,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
        },
        {
            "sku": 10005,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
```


(continues on next page)

(continued from previous page)


```
        },
        {
            "sku": 10006,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
        },
        {
            "sku": 10007,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
        },
        {
            "sku": 10008,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
        },
        {
            "sku": 10009,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
        },
        {
            "sku": 10010,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
        },
        {
            "sku": 10011,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
        },
        {
            "sku": 10012,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
        },
        {
            "sku": 10013,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
        },
        {
            "sku": 10014,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
        },
        {
            "sku": 10015,
```


(continues on next page)

(continued from previous page)


```
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
        },
        {
            "sku": 10016,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
        },
        {
            "sku": 10017,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
        },
        {
            "sku": 10018,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
        },
        {
            "sku": 10019,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
        },
        {
            "sku": 10020,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
        },
        {
            "sku": 10021,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
        },
        {
            "sku": 10022,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
        },
        {
            "sku": 10023,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
        },
        {
            "sku": 10024,
            "weight": "10000",
            "relative_weight": "0.00003862233304",
            "relative_percent": "0.00386223330381"
```


(continues on next page)

(continued from previous page)

```
            }
            ]
        }
        ]
    }
    }
```

1.7.3 ICase/GetMinimumOpenVolume

GET https://api-trade.opskins.com/ICase/GetMinimumOpenVolume/v1

Returns the number of cases required to open in each case-opening request.

Authentication

No auth required.

Input

none

Output

Parameter	Type	Description
count	int	The number of cases required to open in each case-opening request.

1.7.4 ICase/OpenWithKeys

Info

Endpoint is disabled until further notice. (Jan. 11 2019)

POST https://api-trade.opskins.com/ICase/OpenWithKeys/v1

Open a Case with Keys

Authentication

API key required.

OAuth Scopes

`manage_items`

Input

Parameter	Type	Required	Description
case_id	int	•	The ID of the case being opened
amount	int		Number of cases to open. Defaults to `1`. Maximum value of `100`

Output

Parameter	Type	Description
cases	object	Standard OpenedCase Object

1.8 ICaseSite

Warning: Only accounts created with IUser/CreateVCaseUser can access this endpoint.

Endpoints for case websites.

Contents

* *Output*

1.8.1 ICaseSite/GetKeyCount

GET https://api-trade.opskins.com/ICaseSite/GetKeyCount/v1

Returns the number of keys a specific user has on ExpressTrade

Authentication

API key required.

Input

Parameter	Type	Required	Description
trade_url or steam_id	string	•	The trade URL or the Steam ID64 of the user

Output

Parameter	Type	Description
key_count	string	Number of keys this user owns. Parsable into an int.

1.8.2 ICaseSite/GetTradeStatus

GET https://api-trade.opskins.com/ICaseSite/GetTradeStatus/v1

Returns the Trade Status and Opened Case results from an offer created by a case website.

Authentication

API key required.

Input

Parameter	Type	Required	Description
offer_id	int	•	The trade offer ID that was created by the requesting user.

Output

Parameter	Type	Description
offer	object	Standard Trade Offer Object
cases	object	Standard OpenedCase Object

1.8.3 ICaseSite/SendKeyRequest

`POST https://api-trade.opskins.com/ICaseSite/SendKeyRequest/v1`

Sends a trade offer to the user requesting some number of keys for uncasing items on a case website.

Ensure that `remaining_opens` for the case id is greater than 0 via `ICase/GetCaseSchema` , or you will get an HTTP 400 error with the code 314 (TOO_MANY_REDEMPTIONS).

Authentication

API key required.

Note: OPSkins User ID can be found on the WAX ExpressTrade settings page.

Input

Parameter	Type	Required	Description
trade_url or steam_id	string	•	The trade URL or the Steam ID64 of the user
case_id	int	•	The Case ID user wants to open
amount	int		Number of these cases that should be opened. Defaults to 1.
expiration_time	int		Custom expiration time for the trade offer in seconds. Minimum 120 seconds (2 minutes). Defaults to 14 days.
message	string		Trade offer message that will be displayed to the recipient
referral_uid	int		(Optional) You can choose to send this if someone has referred someone else to your site. This should be an OPSkins UID (of the referrer). If this is set, when commission for the cases in this offer is distributed, commission will be split between your site, the referrer, and the case-opening user (rebate, if set below). You may set a custom split rate under ICaseSite/UpdateCommissionSettings with referral_commission_rate. If this is the same as network_user_id in ICaseSite/UpdateCommissionSettings, the referral commission will be merged into the site's commission.
rebate_commission_rate	float		(Optional) You can choose to share commission with the case-opening User. Default `0.00%`, Max `10.00%`, & Min `0.01%`. If this is set, when commission for the cases in this offer is distributed, commission will be split between your site, the User, and referrer (only if set).
			This works similarly to `referral_commission_rate` in ICaseSite/UpdateCommissionSettings

Output

Parameter	Type	Description
offer	object	Standard Trade Offer Object
offer_url	string	Full URL to the trade offer. The recipient of this offer can click this link to view the offer and accept it.

1.8.4 ICaseSite/UpdateCommissionSettings

POST https://api-trade.opskins.com/ICaseSite/UpdateCommissionSettings/v1

Update commission settings for your VCaseUser. Link your OPSkins account to receive commission directly into your USD Wallet.

Authentication

API key required.

Input

Parameter	Type	Required	Description
network_id	int	•	The ID of the network. 1 for OPSkins.com
network_user_id	int	•	User ID on the network. For OPSkins, your OPSkins User ID.
referral_commission_rate	float		(Below)

Important:

This is OPTIONAL!

Want all the commission? Don't worry about this.
This only matters if you send `referral_uid` with `ICaseSite/SendKeyRequest`.

Want to share commission with referrering users?
This property is how many percent commission referrers should receive from total commission percentage (currently 10.00%). Default 5.00%, Max 10.00%, & Min 0.01%.
The 'referrer' is `referral_uid` (OPSkins UID), which can be sent when sending `ICaseSite/SendKeyRequest`. You have to get this from the user (they have to provide to you) and then you can send it.

For example, if this is set to 5.00 %, you will get $0.13 and the referrer $0.12, as the total commission amount (for each case) is 10% of a Skeleton Key, which is $0.25. So by default (5.00%), the referrer will get half the commission $0.12.

Note: You can find your OPSkins User ID on the WAX ExpressTrade settings page.

Output

Parameter	Type	Description
network_id	int	Returns database value (should be same as input).
network_user_id	int	Returns database value (should be same as input).
referral_commission_rate	float	Returns database value (should be same as input).

Listing 10: Output Example

```
{
    "status": 1,
    "time": 1531449864,
    "response": {
        "network_id": 1,
        "network_user_id": 1234567891,
        "referral_commission_rate": 2.5
    }
}
```

1.9 IEthereum

1.9.1 IEthereum/GetContractAddress

GET https://api-trade.opskins.com/IEthereum/GetContractAddress/v1

Returns the most current blockchain transaction hash address for the contract.

Authentication

No auth required.

Input

none

Output

Parameter	Type	Description
contract_address	string	Contract Address Hash

1.10 IItem

Endpoints to handle cases

Contents

1.10.1 IItem/GetAllItems

GET https://api-trade.opskins.com/IItem/GetAllItems/v1/

Authentication

API key required.

Input

Parameter	Type	Required	Description
app_id	int	•	Internal App ID (see ITrade/GetApps)
sku	csv-int		Optional filtering by SKU (for VGO), or def_id for all other items
name	string		Optional filter/search by item market name
page	int		Page number (starting with 1, defaults to 1)
per_page	int		Number of items per page (default 25, max 100, min 1)
sort	int		Standard Item Sorts
no_exclusions	boolean		By default some items are excluded, see list below. Sending 1 here will disable all SKU exclusions.

Hint: Default Excluded SKUs: `VGO 1` (WAX Key)

Output

Parameter	Type	Description
items	array-object	Array of Standard Item Object

Listing 11: Output Example


```
{
    "status":1,
    "time":1538684115,
    "current_page":1,
    "total_pages":9,
    "response":{
       "items":[
          {
             "id":911,
             "sku":10011,
             "wear":0.01472946,
             "pattern_index":362,
             "preview_urls":null,
             "eth_inspect":null,
             "trade_hold_expires":null,
             "internal_app_id":1,
             "inspect":null,
             "tradable":true,
             "attributes":{
                "serial_sku":6,
                "serial_sku_wear":6
             },
             "name":"Bayonet | Poison Target (Factory New)",
             "category":"Covert Knife",
             "rarity":"Covert",
             "type":"Knife",
             "paint_index":null,
             "color":"#eb4b4b",
             "image":{
                "300px":"https://files.opskins.media/file/vgo-img/item/bayonet-
↪poison-target-factory-new-300.png",
                "600px":"https://files.opskins.media/file/vgo-img/item/bayonet-
↪poison-target-factory-new-600.png"
             },
             "suggested_price":14000,
             "suggested_price_floor":14000,
             "wear_tier_index":1
          },
          {
             "id":910,
             "sku":106,
             "wear":0.17418098,
             "pattern_index":769,
             "preview_urls":null,
             "eth_inspect":null,
             "trade_hold_expires":null,
             "internal_app_id":1,
             "inspect":null,
             "tradable":true,
             "attributes":{
                "serial_sku":78,
                "serial_sku_wear":44
             },
             "name":"P90 | Critical (Field-Tested)",
             "category":"Restricted SMG",
             "rarity":"Restricted",
```


(continues on next page)

(continued from previous page)


```
                "type":"SMG",
                "paint_index":null,
                "color":"#8847ff",
                "image":{
                   "300px":"https://files.opskins.media/file/vgo-img/item/p90-critical-
↪field-tested-300.png",
                   "600px":"https://files.opskins.media/file/vgo-img/item/p90-critical-
↪field-tested-600.png"
                },
                "suggested_price":121,
                "suggested_price_floor":121,
                "wear_tier_index":3
             }
          ]
       }
    }
```


1.10.2 IItem/GetItemsById

GET https://api-trade.opskins.com/IItem/GetItemsById/v1/

Get user items by id numbers.

Authentication

API key required.

OAuth Scopes

`items`

Input

Parameter	Type	Required	Description
item_id	int-csv	•	item id filter, separated with comma

Output

Parameter	Type	Description
items	array-object	Array of Standard Item Object
unknown_items	array	Array of item ids that were not found.

1.10.3 IItem/WithdrawToOpskins

POST https://api-trade.opskins.com/IItem/WithdrawToOpskins/v1/

Withdraw items to OPSkins on-site inventory.

Authentication

API key required.

OAuth Scopes

`manage_items`

Input

Parameter	Type	Required	Description
item_id	int-csv	•	item id filter, separated with comma

Output

Parameter	Type	Description
results	object	Result from OPSkins API
output	object	Archived items
-uid	int	OPSkins UID
-items	object	Archived items
–appid	int	Steam App ID
–contextid	int	Steam Context ID
–market_name	string	Market name
–owner_uid	int	OPSkins UID
–wear	float	Wear float value
–original_sale_id	int	Original sale ID on OPSkins

1.10.4 IItem/GetItems

Hint: This endpoint is deprecated in favor of IItem/GetItemDefinitions!

GET or POST https://api-trade.opskins.com/IItem/GetItems/v1/

- Fully supports VGO items
- Partially supports other items for seamless compability with vCase sites.
- (Only if full list of skus is provided from ICase/GetCaseSchema & Beware that the output may contain irrelavent properties to the actual item, such as wear tier)
- All VGO items: `https://api-trade.opskins.com/IItem/GetItems/v1`
- Filter by SKU (VGO only): `https://api-trade.opskins.com/IItem/GetItems/v1?sku_filter=100`
- Filter by SKU & Wear Tier (VGO only): `https://api-trade.opskins.com/IItem/GetItems/v1?sku_filter=100&wear_tier_index=1`

- Multiple SKU (VGO only): `https://api-trade.opskins.com/IItem/GetItems/v1?sku_filter=100,102&wear_tier_index=1`

Note: POST is recommended, as you could easily exceed maximum URI size with GET when using sku_filter. If you are receiving HTTP 500 errors when using GET, this is most likely the reason.

Authentication

None required.

Input

Parameter	Type	Required	Description
sku_filter	int-csv		Optional SKU filter, separated with comma
–wear_tier_index	int		Optional alongside sku_filter

Output

Parameter	Type	Description
items	object	Object containing item meta data
–(sku)	string	SKU number
—-(wear_tier_index)	string	Wear tier index
——(meta data properties)	mix	name, category, rarity, type, color, image, suggested_price, and paint_indexfrom Standard Item Object

Note: VERSION 1 BUG WARNING: For SKU = 1 items (Skeleton Key), the wear tier index is missing. Instead, the key is listed inside of a single-element array. So to access it, you would: *items.1[0].name*. This will be fixed in version 2.

Listing 12: Example Output

```
{
  "status": 1,
  "time": 1524850074,
  "response": {
      "items": {
          "10006": {
              "1": {
                  "name": "Karambit | Poison Target (Factory New)",
                  "category": "Covert Knife",
                  "rarity": "Covert",
                  "type": "Knife",
                  "color": "#eb4b4b",
                  "image": {
                    "300px": "https://files.opskins.media/file/vgo-img/item/karambit-
↪poison-target-factory-new-300.png",
```

(continues on next page)

(continued from previous page)


```
                    "600px": "https://files.opskins.media/file/vgo-img/item/karambit-
↪poison-target-factory-new-600.png"
                },
                "suggested_price": 71436,
                "paint_index": null
            }
        }
    }
}
```


VGO Wear Tier Index Map

These mappings will never change, you may store and use as you please. Items without tiers, e.g. keys, have a wear tier index of *0*.

Listing 13: Example Output


```
{
  "wear_tier_index_map": {
     "": 0,
     "Factory New": 1,
     "Minimal Wear": 2,
     "Field-Tested": 3,
     "Well-Worn": 4,
     "Battle-Scarred": 5
  },
  "wear_tier_index_to_float_map": {
     "1": {
        "min": 0,
        "max": 0.06999999999999
     },
     "2": {
        "min": 0.07,
        "max": 0.14999999999999
     },
     "3": {
        "min": 0.15,
        "max": 0.37999999999999
     },
     "4": {
        "min": 0.38,
        "max": 0.44999999999999
     },
     "5": {
        "min": 0.45,
        "max": 1
     }
  }
}
```


1.10.5 IItem/GetItemDefinitions

GET or POST https://api-trade.opskins.com/IItem/GetItemDefinitions/v1/

- All items for an app (limit `1000` per page): `GetItemDefinitions/v1?app_id=1`
- Filter by `def_id`: `GetItemDefinitions/v1?app_id=1&def_id_filter=900000001,900000002`

Note: POST is recommended, as you could easily exceed maximum URI size with GET when using sku_filter. If you are receiving HTTP 500 errors when using GET, this is most likely the reason.

Authentication

None required.

Input

Parameter	Type	Required	Description
app_id	int	•	Internal App ID (see ITrade/GetApps)
def_id_filter	csv-int		Optional def_id comma-separated filter
index_by	string		Optionally index the output by market_name, def_id, or sku, send it as literal string
page	int		Page number in response (starting with 1, defaults to 1)
per_page	int		Number of items per_page in response (no more than 1000 (default))

Output

Parameter	Type	Description
definitions	array-object or object	An array of objects or object list if index_by option is used
–def_id	int	Unique Definition ID, this is a unique & unchanging identifier for each item, regardless of app_id. Not to be confused with sku, which is not unique per wear-tierfor VGO items. VGO item def_id starts at 900,000,000 for no particular reason.
–sku	int	SKU for item. Mainly utilized for VGO items, for all other items, this will be the same as def_id.
–internal_app_id	int	Internal App ID
–name	string	Name, non-unique, most likely the same as market_name however
–market_name	string	Market name, unique per app_id
–color	string	Color with hex # for VGO (ID 1), for all others, no # – usually corresponds to the rarity of the item
–image	string	Generic image URL
–suggested_price	int	Market suggested price
–suggested_price_floor	int	The minimum viable suggested price, does not change.
–attributes	object	Generic (non-unique) item attributes, all app-specific properties will be in here

Listing 14: Output Example (Array of objects)

```
{
    "status":1,
    "time":1544467201,
    "current_page":1,
    "total_pages":1,
    "response":{
        "definitions":[
            {
                "def_id":900000001,
                "internal_app_id":1,
                "name":"WAX Key",
                "market_name":"WAX Key",
                "color":"#777777",
                "image":"https://files.opskins.media/file/vgo-img/item/wax-key-300.png",
                "suggested_price":250,
                "suggested_price_floor":250,
                "attributes":{
                    "category":"WAX Key",
                    "image_generic_300":"https://files.opskins.media/file/vgo-img/item/
↪wax-key-300.png",
                    "image_generic_600":"https://files.opskins.media/file/vgo-img/item/
↪wax-key-600.png",
                    "image_generic_900":"https://files.opskins.media/file/vgo-img/item/
↪wax-key-900.png",
```

(continues on next page)

(continued from previous page)


```
                "image_generic_1800":"https://files.opskins.media/file/vgo-img/item/
↪wax-key-1800.png",
                "image_generic_2500":"https://files.opskins.media/file/vgo-img/item/
↪wax-key-2500.png",
                "paint_index":null,
                "rarity":null,
                "suggested_price_floor":250,
                "type":"WAX Key",
                "wear_tier_index":0
            }
        }
    ]
  }
}
```


Listing 15: Output Example (Indexed by `def_id`)


```
{
   "status":1,
   "time":1544467222,
   "current_page":1,
   "total_pages":1,
   "response":{
      "definitions":{
         "900000001":{
            "def_id":900000001,
            "internal_app_id":1,
            "name":"WAX Key",
            "market_name":"WAX Key",
            "color":"#777777",
            "image":"https://files.opskins.media/file/vgo-img/item/wax-key-300.png",
            "suggested_price":250,
            "suggested_price_floor":250,
            "attributes":{
               "category":"WAX Key",
               "image_generic_300":"https://files.opskins.media/file/vgo-img/item/
↪wax-key-300.png",
               "image_generic_600":"https://files.opskins.media/file/vgo-img/item/
↪wax-key-600.png",
               "image_generic_900":"https://files.opskins.media/file/vgo-img/item/
↪wax-key-900.png",
               "image_generic_1800":"https://files.opskins.media/file/vgo-img/item/
↪wax-key-1800.png",
               "image_generic_2500":"https://files.opskins.media/file/vgo-img/item/
↪wax-key-2500.png",
               "paint_index":null,
               "rarity":null,
               "suggested_price_floor":250,
               "type":"WAX Key",
               "wear_tier_index":0
            }
         },
         "900000002":{
            "def_id":900000002,
            "internal_app_id":1,
            "name":"AK-47 | Overdrive (Factory New)",
```


(continues on next page)

(continued from previous page)


```
            "market_name":"AK-47 | Overdrive (Factory New)",
            "color":"#eb4b4b",
            "image":"https://files.opskins.media/file/vgo-img/item/ak-47-overdrive-
↪factory-new-300.png",
            "suggested_price":23252,
            "suggested_price_floor":23252,
            "attributes":{
               "category":"Covert Rifle",
               "image_generic_300":"https://files.opskins.media/file/vgo-img/item/
↪ak-47-overdrive-factory-new-300.png",
               "image_generic_600":"https://files.opskins.media/file/vgo-img/item/
↪ak-47-overdrive-factory-new-600.png",
               "image_generic_900":"https://files.opskins.media/file/vgo-img/item/
↪ak-47-overdrive-factory-new-900.png",
               "image_generic_1800":"https://files.opskins.media/file/vgo-img/item/
↪ak-47-overdrive-factory-new-1800.png",
               "image_generic_2500":"https://files.opskins.media/file/vgo-img/item/
↪ak-47-overdrive-factory-new-2500.png",
               "paint_index":null,
               "rarity":"Covert",
               "suggested_price_floor":23252,
               "type":"Rifle",
               "wear_tier_index":1
            }
         }
      }
   }
}
```


1.10.6 IItem/GetRarityStats

GET https://api-trade.opskins.com/IItem/GetRarityStats/v1/

Get item rarity stats per Definition ID (SKU) (currently only for VGO)

Authentication

API key required.

Input

Parameter	Type	Required	Description
app_id	int	•	Internal App ID (see ITrade/GetApps)
def_id	int-csv		Definition IDs (SKUs) separated by commas

- If an item was never unboxed (very rare items), no stats will be outputted
- An individual item's permanent serial number will be inside Standard Item Object as `serial_sku_wear`.

Output

Parameter	Type	Description
items	object	Object containing rarity data per Definition ID
-(def_id)	string	Definition ID
–def_id	int	Definition ID
–def_sub_id	int/null	Sub-Definition ID, for VGO this is the Wear Tier Index (1,2,3,4,5)
–latest_serial	int	The latest Serial Number given for an item of this type (only per Def ID). Not currently displayed on our sites.
–sub_items	object	Object containing rarity data per Definition ID & Sub Definition ID
—(def_sub_id)	string	Sub-Definition ID
—-def_id	int	Definition ID
——def_sub_id	int	Sub-Definition ID
——latest_serial	int	The latest serial number given for an item of this type. This is what is displayed as "Total Unboxed" on WAX ExpressTrade & OPSkins Marketplace.

Listing 16: Output Example


```
{
   "status":1,
   "time":1536707797,
   "response":{
      "items":{
         "102":{
            "def_id":102,
            "def_sub_id":null,
            "latest_serial":2,
            "sub_items":{
               "2":{
                  "def_id":102,
                  "def_sub_id":2,
                  "latest_serial":1
               },
               "5":{
                  "def_id":102,
                  "def_sub_id":5,
                  "latest_serial":1
               }
            }
         }
      }
   }
}
```


1.10.7 IItem/InstantSellRecentItems

POST https://api-trade.opskins.com/IItem/InstantSellRecentItems/v1/

This endpoint can be used to instant-sell recently (15 min) unboxed items on OPSkins. Items are automatically transferred to OPSkins and then sold via the endpoint ISales/InstantSellItems/v1. Note that partial success is possible

with this endpoint. It's also possible that we will send a `status` of `1` but the OPSkins endpoint will fail completely, as shown in the Output Examples below.

Authentication

API key required.

OAuth Scopes

`instant_sell_recent_items`

- If using OAuth, OPSkins wallet balance information will not be shown unless balance scope is available.

Allowed Apps

- All apps & items allowed. Note that the OPSkins endpoint may still reject some apps & items.

Input

Parameter	Type	Required	Description
item_id	int-csv	•	List of Item IDs, separated with commas. Maximum 100.
instant_sell_type	int		1 for OPSkins Credits, 2 for USD (default)

Output

Parameter	Type	Description
valid_item_ids	array-int	Item IDs considered valid
unknown_item_ids	array-int	Item IDs that were not found in the database or do not belong to you
not_recent_item_ids	array-int	Item IDs created more than 15 minutes ago, which are not eligible
ineligible_item_ids	array-int	Item IDs that are currently not eligible for trade or transfer
not_allowed_item_ids	array-int	Deprecated (all apps & items allowed). Item IDs that are not allowed for this endpoint. See Allowed Apps above.
isales_instantsellitems_v1	mixed	Full ISales/InstantSellItems/v1 response from OPSkins API

Listing 17: Output Example (partial success)

```
{
   "status":1,
   "time":1542928287,
   "response":{
      "valid_item_ids":[
```

(continues on next page)

(continued from previous page)


```
            391
        ],
        "unknown_item_ids":[
            291,
            292
        ],
        "not_recent_item_ids":[

        ],
        "ineligible_item_ids":[

        ],
        "not_allowed_item_ids":[

        ],
        "isales_instantsellitems_v1":{
            "status":1,
            "time":1542928287,
            "balance":500027520,
            "credits":245,
            "cryptoBalances":{
                "ETH":"0.000000000000000000",
                "WAX":"0.000000000000000000"
            },
            "response":{
                "items":[
                    {
                        "saleid":309421537,
                        "new_itemid":309421538,
                        "item_id":391,
                        "name":"Huntsman Knife | Cyber Sport (Battle-Scarred)"
                    }
                ],
                "items_count":1,
                "total_value":{
                    "usd":6601,
                    "credits":0
                }
            }
        }
    }
```


Listing 18: Output Example (None of the provided items exist or belong to you)


```
{
    "status": 312,
    "time": 1542910778,
    "message": "None of the items provided exist or belong to you: 159, 160"
}
```

Listing 19: Output Example (None of the items provided are valid/eligible)


```
{
   "status": 312,
   "time": 1542910641,
   "message": "None of the items provided are valid/eligible",
   "response": {
      "valid_item_ids": [],
      "unknown_item_ids": [
            159,
            160
      ],
      "not_recent_item_ids": [
            180
      ],
      "ineligible_item_ids": [],
      "not_allowed_item_ids": []
   }
}
```


Listing 20: Output Example (OPSkins API Error)


```
{
   "status": 1,
   "time": 1542852394,
   "response": {
      "isales_instantsellitems_v1": {
            "status": 2000,
            "time": 1542852394,
            "message": "Something went wrong."
      }
   }
}
```

Listing 21: Output Example (Error while transferring items to OPSkins)

```
{
    "status": 202,
    "time": 1542864143,
    "message": "Error during transfer of items to OPSkins. It's possible the items␣
↪were transferred successfully.",
    "response": {
        "valid_item_ids": [
            174
        ],
        "unknown_item_ids": [
            159,
            160
        ],
        "not_recent_item_ids": [],
        "ineligible_item_ids": [],
        "not_allowed_item_ids": []
    }
}
```

1.11 ITrade

Endpoints which allows Peer-to-Peer Trading.

Contents

1.11.1 ITrade/AcceptOffer

POST https://api-trade.opskins.com/ITrade/AcceptOffer/v1/

Accepts offer sent by another user

Authentication

API key required.

OAuth Scopes

- `trades`
- `open_cases` (restricted to only case-opening offers)

Input

Parameter	Type	Required	Description
twofactor_code	string	•	2-factor authentication code
offer_id	int	•	Trade offer Id you want to accept

Output

Parameter	Type	Description
offer	object	Standard Trade Offer Object
new items	array-object	New items for the recipient (user that makes this API call). Standard Item Object
failed_cases	int	A count of failed cases opened with keys, 0 if none failed.

1.11.2 ITrade/CancelOffer

POST https://api-trade.opskins.com/ITrade/CancelOffer/v1/

Cancels a trade offer

If cancelled by the sender it will go into *STATE_CANCELLED* (6) and if cancelled by the receiver it will go into *STATE_DECLINED* (7).

Authentication

API key required.

OAuth Scopes

`trades`

Input

Parameter	Type	Required	Description
offer_id	int	•	Offer ID that you're a party to (sender or receiver)

Output

Parameter	Type	Description
offer	object	Standard Trade Offer Object

1.11.3 ITrade/GetApps

`GET https://api-trade.opskins.com/ITrade/GetApps/v1/`

Get all supported apps and their descriptions.

Authentication

No auth required.

Input

none

Output

Parameter	Type	Description
apps	object	List of apps and descriptions
–internal_app_id	int	Internal App ID
–steam_app_id	int	Steam App ID
–steam_context_id	int	Steam Context ID
–name	string	Short name of app
–long_name	string	Long name of app
–img	string	Image URL of app icon https://opskins.com/images/games/logo-small-vgo.jpg
–img_thumb	string	Thumbnail image for app https://opskins.com/images/game-thumb-vgo.jpg
–default	int	If property exists, this is the default app. Not outputted for other apps.

Listing 22: Output Example

```
{
    "status": 1,
    "time": 1528135996,
    "response": {
        "apps": [
            {
                "internal_app_id": 1,
                "steam_app_id": 1912,
                "steam_context_id": 1,
                "name": "VGO",
                "long_name": "VGO",
                "img": "https://opskins.com/images/games/logo-small-vgo.jpg",
                "default": 1
            }
        ]
    }
}
```

1.11.4 ITrade/GetOffer

GET https://api-trade.opskins.com/ITrade/GetOffer/v1/

Get an individual trade offer

You must be one of the parties involved in the offer (sender/receiver).

Authentication

API key required.

OAuth Scopes

`items`

Input

Parameter	Type	Required	Description
offer_id	int	•	ID of trade offer

Output

Parameter	Type	Description
offer	object	Standard Trade Offer Object

1.11.5 ITrade/GetOffers

GET https://api-trade.opskins.com/ITrade/GetOffers/v1/

Get user's trade offers

Authentication

API key required.

OAuth Scopes

`items`

Input

Parameter	Type	Required	Description
uid	int		ID of other user, involved in offers
state	string		A comma-separated list of offer states to filter by (See available states in ITrade).
type	string		One of sent, received
page	int		page number in response (starting with 1, default to 1)
per_page	int		number of items per_page in response (no more than 100, defaults to 100)
ids	int-csv		Trade offer IDs list filter
sort	string		One of created, expired, modified

Output

Parameter	Type	Description
offers	array-object	Array of Standard Trade Offer Object
total	int	Total number of offers matching the input filters

1.11.6 ITrade/GetTradeUrl

GET https://api-trade.opskins.com/ITrade/GetTradeURL/v1/

Get your account's trade URL, allowing P2P trading.

Authentication

API key required.

OAuth Scopes

`identity_basic`, `identity`, `trades`

Input

none

Output

Parameter	Type	Description
uid	int	Your OPSkins User ID
token	string	Your trade token
long_url	string	The actual URL someone should go to in order to send a trade offer to your account.
short_url	string	A shortened alias for long_url of the type "…/t/1/Lhn9d7fVL1U". This redirects to the long URL.

1.11.7 ITrade/GetUserInventory

`GET https://api-trade.opskins.com/ITrade/GetUserInventory/v1/`

Get trade offer recipient's inventory.

Authentication

No auth required.

Input

Parameter	Type	Required	Description
uid	int	•	User ID of user whose inventory you want to see
app_id	int	•	Internal App ID (see ITrade/GetApps)
page	int		Page number in response (starting with 1, defaults to 1)
per_page	int		Number of items per_page in response (no more than 500)
search	string		Additional search by item's name
sort	int		Code to set how results should be sorted. See available types below or in the output response

Output

Parameter	Type	Description
total	int	Total number of items (filtered, if search parameter is passed)
items	object	Standard Item Object
user_data	object	Standard User Public Profile Object
sort_parameters	array-object	Available sort parameters

Sort parameter values

- *1*: By name ASC (alphabetical, *z* first)
- *2*: By name DESC (alphabetical, *a* first)
- *3*: By last_update ASC (oldest first)
- *4*: By last_update DESC (newest first)
- *5*: By suggested price ASC (lowest first)
- *6*: By suggested price DESC (highest first)

1.11.8 ITrade/GetUserInventoryFromSteamId

GET https://api-trade.opskins.com/ITrade/GetUserInventoryFromSteamId/v1/

Get trade offer recipient's inventory by SteamID.

Authentication

API key required.

OAuth Scopes

`items`, `trades`

Input

Parameter	Type	Required	Description
steam_id	int	•	Steam ID of user whose inventory you want to see
app_id	int	•	Internal App ID (see ITrade/GetApps)
page	int		Page number in response (starting with 1, default to 1)
per_page	int		Number of items per_page in response (no more than 500)
search	string		Additional search by item's name
sort	int		Code to set how results should be sorted. See available types below or in the output response

Output

Parameter	Type	Description
total	int	Total number of items (filtered, if search parameter is passed)
items	object	Standard Item Object
user_data	object	Standard User Public Profile Object
sort_parameters	array-object	Available sort parameters

Sort parameter values

- *1*: By name ASC (alphabetical, *z* first)
- *2*: By name DESC (alphabetical, *a* first)
- *3*: By last_update ASC (oldest first)
- *4*: By last_update DESC (newest first)
- *5*: By suggested price ASC (lowest first)
- *6*: By suggested price DESC (highest first)

1.11.9 ITrade/RegenerateTradeUrl

`POST https://api-trade.opskins.com/ITrade/RegenerateTradeURL/v1/`

Regenerate your account's trade URL for P2P trading, invalidating the old one.

Authentication

API key required.

OAuth Scopes

`edit_account`

Input

none

Output

Parameter	Type	Description
uid	int	Your OPSkins User ID
token	string	Your new trade token
long_url	string	The actual URL someone should go to in order to send a trade offer to your account.
short_url	string	A shortened alias for long_url of the type "…/t/1/Lhn9d7fVL1U". This redirects to the long URL.

1.11.10 ITrade/SendOffer

`POST https://api-trade.opskins.com/ITrade/SendOffer/v1/`

Sends trade offer to another user including your and their items

Authentication

API key required.

OAuth Scopes

`trades`

Input

One of: `uid` + `token` **or** `trade_url` is required.

Parameter	Type	Required	Description
twofactor_code	int	•	2-factor authentication code
uid	int		User ID of user you want to send your trade offer to
token	string		Trade token of user you want to send your trade offer to
trade_url	string		Trade URL of the user you want to send your trade offer to.
items_to_send	csv-int		A comma-separated list of (int) Item IDs you wish to send to recipient. Maximum 100 items.
items_to_receive	csv-int		A comma-separated list of (int) Item IDs you wish to receive from the recipient. Maximum 100 items.
expiration_time	int		Custom expiration time for an offer in seconds. Minimum 120 seconds (2 minutes). Defaults to 14 days.
message	string		Trade offer message that will be displayed to the recipient

Output

Parameter	Type	Description
offer	object	Standard Trade Offer Object

1.11.11 ITrade/SendOfferToSteamId

POST https://api-trade.opskins.com/ITrade/SendOfferToSteamId/v1/

Sends trade offer to another user, including your and their items

Authentication

API key required.

OAuth Scopes

`trades`

Input

Parameter	Type	Required	Description
twofactor_code	int	•	2FA Auth Code
steam_id	string	•	Steam ID of user you want to send your trade offer to
items_to_send	csv-int		A comma-separated list of (int) Item IDs you wish to send to recipient. Maximum 100 items.
items_to_receive	csv-int		A comma-separated list of (int) Item IDs you wish to receive from the recipient. Maximum 100 items.
expiration_time	int		Custom expiration time for an offer in seconds. Minimum 120 seconds (2 minutes). Defaults to 14 days.
message	string		An optional message to include with your trade offer, up to 190 characters.

Output

Parameter	Type	Description
offer	object	Standard Trade Offer Object

1.12 IUser

Endpoints to handle user related things

Contents

1.12.1 IUser/CreateVCaseUser

`POST https://api-trade.opskins.com/IUser/CreateVCaseUser/v1/`

Create a special case-website user

VCase Site users are restricted from most parts of the API. They cannot own items or send regular trades. But they gain access to a set of new API endpoints under the ICaseSite interface.

You generally only need to create a VCase Site API key once, which can be done with the following example CURL command.

```
$ curl -d '{"site_url":"http://yoursite.com","display_name":"yoursite"}' -H "Content-
↪Type: application/json" -X POST https://api-trade.opskins.com/IUser/CreateVCaseUser/
↪v1/
```

Authentication

No auth required.

Creates a case-website user, which can access all endpoints under the ICaseSite interface.

Input

Parameter	Type	Required	Description
site_url	string	•	Must be a valid & unique full URL to your case-website.
display_name	string	•	Display name for case-website user. This name will appear in all trade offers.

Output

Parameter	Type	Description
api_key	string	User API key. Keep it in a safe place and use it to access ICaseSite endpoints.
user	object	Standard User Profile Object – User ID will not be a matching OPSkins UID, it will be unique to WAX ExpressTrade and will be a negative integer.

1.12.2 IUser/GetInventory

`GET https://api-trade.opskins.com/IUser/GetInventory/v1/`

Get Your Inventory

Authentication

API key required.

OAuth Scopes

`items`, `trades`

Input

Parameter	Type	Required	Description
app_id	int	•	Internal App ID (see ITrade/GetApps)
page	int		Page number in response (starting with 1, defaults to 1)
per_page	int		Number of items per_page in response (no more than 500)
search	string		Additional search by item's name
sort	int		Standard Item Sorts
filter_in_trade	boolean		Removes items that are part of an active trade from the response.

Output

Parameter	Type	Description
total	int	Total number of items (filtered, if search parameter is passed)
items	array-object	Items list, based on pagination and search filters. Standard Item Object
sort_parameters	array-object	Available sort parameters
items_in_active_offers	object-array	List of Item IDs and matching Offer IDs that are involved in active trade offers. Keys are Item IDs and values are an array of Offer IDs.
–value	int	Value expected in this method
–display_name	string	Display name

1.12.3 IUser/GetProfile

GET https://api-trade.opskins.com/IUser/GetProfile/v1/

Get Your Profile

Authentication

API key required.

Authentication

`identity_basic`, `identity`

Input

Parameter	Type	Required	Description
with_extra	bool		Should we send sensitive user data? Defaults to false

Output

Parameter	Type	Description
user	object	Standard User Profile Object

1.12.4 IUser/UpdateProfile

POST https://api-trade.opskins.com/IUser/UpdateProfile/v1/

Update Your Profile

Authentication

API key required.

OAuth Scopes

`edit_account`

Input

Parameter	Type	Required	Description
display_name	string		Name to display on trade offers
inventory_is_private	boolean		Whether inventory is private (nobody can see it, even with token)
allow_twofactor_code_reuse	boolean		Allow Two Factor code reuse for certain features (Send Offer, Accept Offer)
auto_accept_gift_trades	boolean		Auto-accept gift trade offers
anonymous_transactions	boolean		Hide my username in WAX transaction records

Output

Parameter	Type	Description
user	object	Standard User Profile Object

1.12.5 IUser/UserReports

POST https://api-trade.opskins.com/IUser/UserReports/v1

Authentication

API key required.

Input

Parameter	Type	Required	Description
message	string	•	Message included in the report
report_type	integer	•	Reason - spam = 1, phishing = 2, error = 3;
offer_id	integer	•	Id of the reported offer

Output

Parameter	Type	Description
success	boolean	true if everything went well

1.13 Additional Information

Contents

1.13.1 Standard OpenedCase Object

Parameter	Type	Description
id	int	Unique Case ID
status	int	Case Status ID
status_text	string	Case Status Description
case_id	int	Case Schema ID
case_site_trade_offer_id	int	Trade Offer ID
item	object	Standard Item Object

1.13.2 Case Status

- STATE_ERROR = *1*
- STATE_PENDING = *2*
- STATE_OPENED = *3*

Listing 23: Output Example for Standard OpenedCase Object

```
{
   "status":1,
   "time":1535545650,
   "response":{
     "id": 1,
     "status": 3,
     "status_text": "Opened",
     "case_id": 1,
     "case_site_trade_offer_id": 3215,
     "item": { }
   }
}
```

1.13.3 Standard Item Object

Parameter	Type	Description
id	int	Item ID
internal_app_id	int	Internal App ID (see ITrade/GetApps)
sku	int	Item definition (meta-data) SKU #
wear	float	Wear float value, only applicable for certain apps
trade_hold_expires	int / null	Trade hold expiration date. null if no trade hold
name	string	Market name e.g. "MAG-7 Gold Digger (Factory New)"
category	string	Category name e.g. "Restricted Rifle"
rarity	string	Category rarity e.g. "Restricted" – only outputted for VGO
type	string	Category type e.g. "Rifle" – only outputted for VGO
color	string	Color hex, includes #
image	object	Generic image URLs
–300px	string	300px image URL - https://files.opskins.media/file/vgo-img/item/dual-berettas-trigger-happy-battle-scarred-300.png
–600px	string	600px image URL - https://files.opskins.media/file/vgo-img/item/dual-berettas-trigger-happy-battle-scarred-600.png
suggested_price	int	OPSkins 7-day suggested price (US cents)
suggested_price_floor	int	(Only for VGO) The minimum viable suggested price, does not change.
preview_urls	object	Field Inspection URLs for VGO items. Some of these properties may not be outputted if not available. If they are provided, the image or video itself may not be generated yet, so you should fallback to generic images provided in image object.
–thumb_image	string	https://files.opskins.media/file/vgo-img/previews/163609_thumb.jpg
–front_image	string	https://files.opskins.media/file/vgo-img/previews/163638_front.jpg
–back_image	string	https://files.opskins.media/file/vgo-img/previews/163638_back.jpg
–video	string	https://files.opskins.media/file/vgo-img/previews/163638_video.webm
inspect	string / null	Steam in-game inspection URL. Can be null.
eth_inspect	string / null	Etherscan.io Ethereum Transaction URL. null for inapplicable apps.
pattern_index	int	Pattern index (value between 1-1000) (only available for VGO, null for other apps)
paint_index	int / null	Paint index value for a CS:GO item. 0 or null for items without a paint-index.

1.13.4 Output Example for Standard Item Object

Listing 24: Output Example for Standard Item Object

```
{
   "id":363645,
   "sku":10006,
```

(continues on next page)

(continued from previous page)


```
   "wear":0.583130179639101,
   "pattern_index":549,
   "preview_urls":{
      "thumb_image":"https://files.opskins.media/file/vgo-img/previews/164325_thumb.
↪jpg",
      "front_image":"https://files.opskins.media/file/vgo-img/previews/164342_front.
↪jpg",
      "back_image":"https://files.opskins.media/file/vgo-img/previews/164342_back.jpg
↪",
      "video":"https://files.opskins.media/file/vgo-img/previews/164342_video.webm"
   },
   "eth_inspect":null,
   "trade_hold_expires":null,
   "internal_app_id":1,
   "inspect":null,
   "name":"Karambit | Poison Target (Factory New)",
   "category":"Covert Knife",
   "rarity":"Covert",
   "type":"Knife",
   "paint_index":null,
   "color":"#eb4b4b",
   "image":{
      "300px":"https://files.opskins.media/file/vgo-img/item/karambit-poison-target-
↪factory-new-300.png",
      "600px":"https://files.opskins.media/file/vgo-img/item/karambit-poison-target-
↪factory-new-600.png"
   },
   "suggested_price":71436
}
```

1.13.5 Standard Trade Offer Object

Parameter	Type	Description
offer	object	Holds offer and item data
–id	int	offer id
–sender	object	Offer sender's information
—-uid	int	Sender's uid
—-steam_id	string	Senders's SteamID
—-display_name	string	Sender's display name
—-avatar	string	Sender's avatar image url
—-verified	bool	Is this user verified on OPSkins by support?
—-items	object	Items which sender offered for trade in the offer. Standard Item Object
–recipient	object	Offer recipient's information
—-uid	int	Recipient's uid
—-steam_id	string	Recipient's SteamID
—-display_name	string	Recipient's display name
—-avatar	string	Recipient's avatar image url
—-verified	bool	Is this user verified on OPSkins by support?
—-items	object	Recipient's items which sender wanted to receive in the offer. Standard Item Object
–state	int	Offer state int – Offer States
–state_name	string	State's display name e.g "Active"
–time_created	int	Offer creation unix timestamp
–time_updated	int	Last update unix timestamp
–time_expires	int	Offer expiration unix timestamp
–message	string	Message from sender to receiver
–is_gift	boolean	Whether or not this offer is a gift (you are not losing any items).
–is_case_opening	boolean	Whether or not this offer is from a vCase website.
–sent_by_you	bool	Whether or not the offer was sent by you. Not outputted on no-auth endpoints.

1.13.6 Offer States

- **STATE_ACTIVE = *2***
  - The offer is active and the recipient can accept it to exchange the items
- **STATE_ACCEPTED = *3***
  - The recipient accepted the offer and items were exchanged
- **STATE_EXPIRED = *5***
  - The offer expired from inactivity
- **STATE_CANCELED = *6***
  - The sender canceled the offer
- **STATE_DECLINED = *7***
  - The recipient declined the offer
- **STATE_INVALID_ITEMS = *8***
  - One of the items in the offer is no longer available so the offer was canceled automatically
- **STATE_PENDING_CASE_OPEN = *9***

  - The trade offer was initiated by a VCase site and it's awaiting eth confirmations. User's keys have been removed, but may be restored on error later.

- **STATE_EXPIRED_CASE_OPEN = *10***
  - The trade offer was initiated by a VCase site and there was an error opening case due to back-end issues. No items should have been exchanged.
- **STATE_FAILED_CASE_OPEN = *12***
  - The trade offer was initiated by a VCase site and we were unable to generate items on the blockchain, so the user's keys have been refunded.

If a case opening succeeds from a vcase site, the offer will go into *STATE_ACCEPTED* and the items generated from the case opening will appear in the trade offer as if they came from the vcase site user. The end-result is that the user will see their keys exchanged for items in the trade offer on success.

1.13.7 Standard User Profile Object

Parameter	Type	Description
user	object	Holds user info
–id	int	OPSkins.com User ID
–steam_id	string	Steam ID64
–display_name	string	Display name
–avatar	string	URL to avatar
–twofactor_enabled	boolean	Whether or not user has Two-Factor Auth enabled.
–api_key_exists	boolean	See whether user has API Key
–sms_phone	string/null	(Optional via with_extra) Phone number used for SMS verification
–contact_email	string/null	(Optional via with_extra) Email address
–inventory_is_private	boolean	(Optional via with_extra) Set whether inventory is private (nobody can see it, even with token)
–allow_twofactor_code_reuse	boolean	Allow Two Factor code reuse for certain features (Send Offer, Accept Offer)

1.13.8 Standard User Public Profile Object

Parameter	Type	Description
user_data	object	Holds user info
–username	string	Display name
–avatar	string	URL to avatar