Table of Contents

What is it?

Topaz is a service that enables the sharing of real-time data to interested parties via a modern architecture and delivering the data in a modern data format.

Topaz is not a new Data Distribution Center (DDC) in that it does not perform the same functionality. The DDC offered data in a variety of formats, such as CSV and XML, and delivered that data in the form of a file that needed to be picked up from an SFTP server. It was prone to failures and by the time the file was written to the server, it was potentially already out of date.

Topaz, instead, is an API Service that serves up real time data on request and can be used in real-time systems. it can be integrated with any system that can call an API and as stated, serves data in a modern JSON format.

Who is it for?

Anyone that needs or wants access to our data and anyone that has permission to use the other jurisdictions data.

GRV: GRV Staff are able to use the Topaz API in any of our applications and in fact, the new NDX uses Topaz and the OnTrack App also uses Topaz.
Punters: These are people that bet on the dogs and that collect our data, insert that into their own systems to make educated decisions on which dogs to bet on. Punters are able to gain access to Victorian data free of charge and begin using the API.
Jurisdictions: The other jurisdictions are able to gain access to the data we store about their races and use that data in their own systems. In fact, GRNSW and GRSA are currently doing that with their web pages.
Corporations: These are companies that use our data and then might publish our meetings and results in media such as podcasts and print.
Anyone else: Anyone that applies for a Topaz account is granted access to Victorian data by default and can begin using the system.

Location

Topaz client portal is located at https://topaz.grv.org.au/. There are Development, SIT and UAT sites that are not covered in this document.

Home Screen

The Topaz home screen only gives you access to the API documentation, which you need an API Key to use, as well as the history of versions and changes within those versions.

Its only purpose is to serve as a place to store versions, link to the documentation and link to the login facility.

Logging In

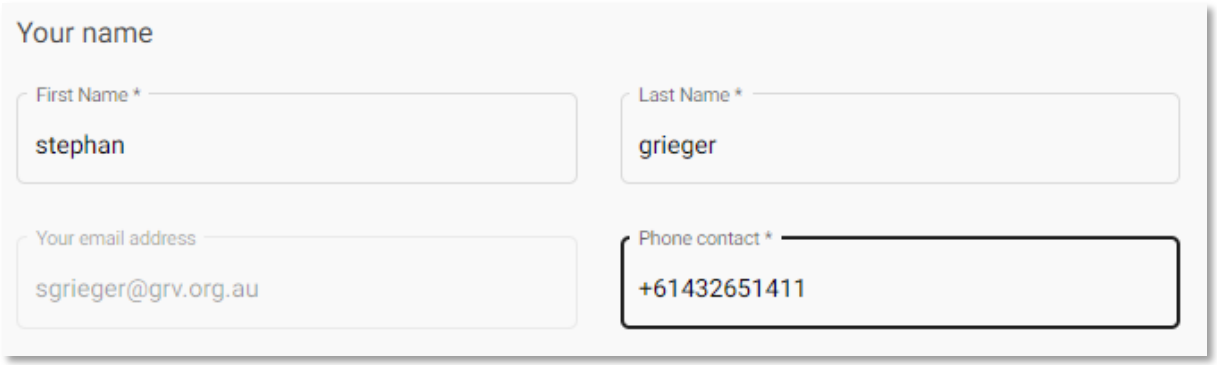
You must have a valid and current login for Topaz or you need to create an account for yourself. The process for this is to press on the Login button in the top right of the home page. This button has the functionality to both log you in or take you to a sign-up page.

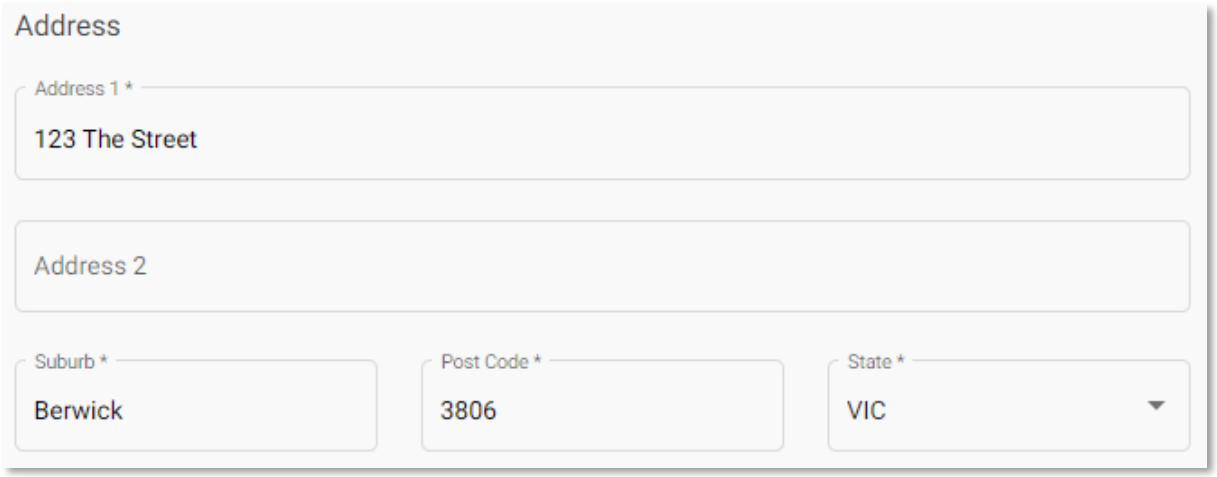
If you have forgotten your password, the login page also allows you to reset your password by sending you a reset password link to your email address.

Profile

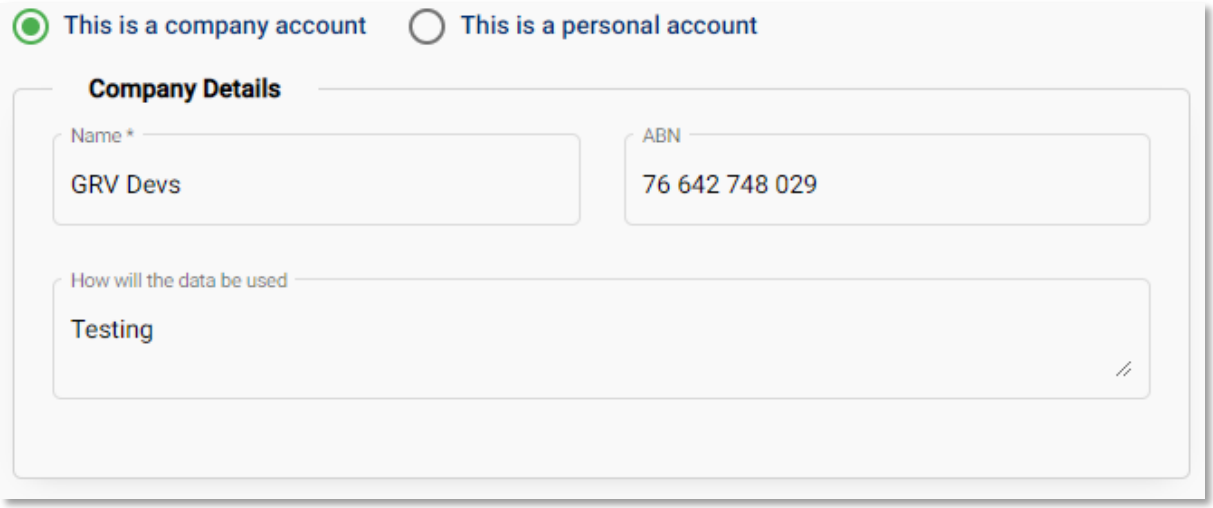
The profile page provides GRV with a minimum viable amount of information about the account holder, their intended use of the data and whether they are a private or corporate entity. GRV has attempted to keep the level of information about our users to a minimum.

We hold personal information such as the account holders name, phone, email and address.





Past that the only other information required is some basic usage details and the ABN if this is a corporate account. For private accounts, no more information is required or captured.

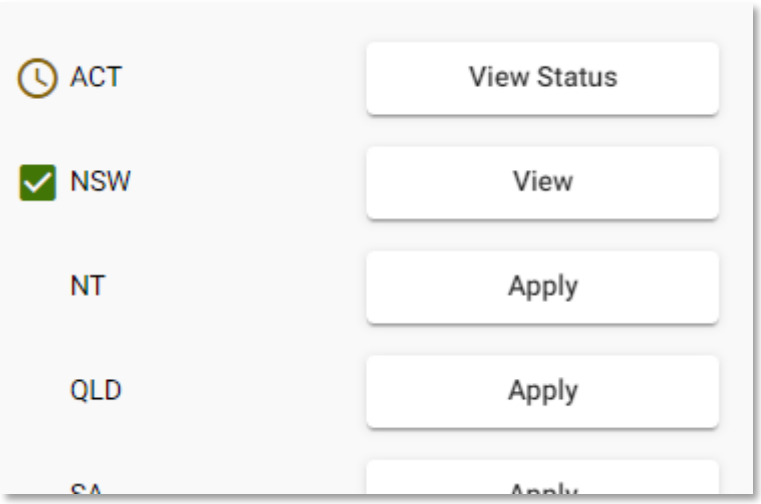


Data Applications

Users and Corporations that create accounts are automatically given access to Victorian data and the requirement to provide supporting documentation is removed. However, for anyone wanting to gain access to other jurisdictions data, they must apply for that data and provide documentation from that jurisdiction that they are authorised to obtain that data.

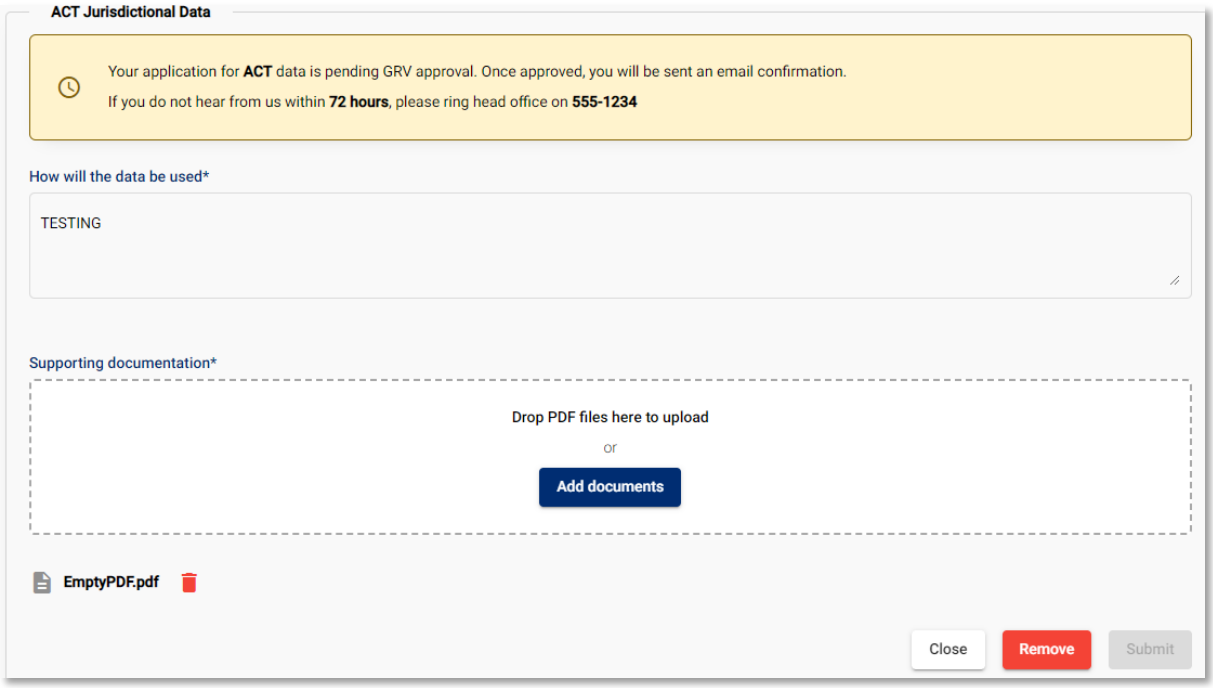
To do this, the user must have already received digital authorisation from the jurisdiction that they can upload to the portal.

To apply for additional data through the Topaz portal, click on the [Data Application] menu at the top of the screen.



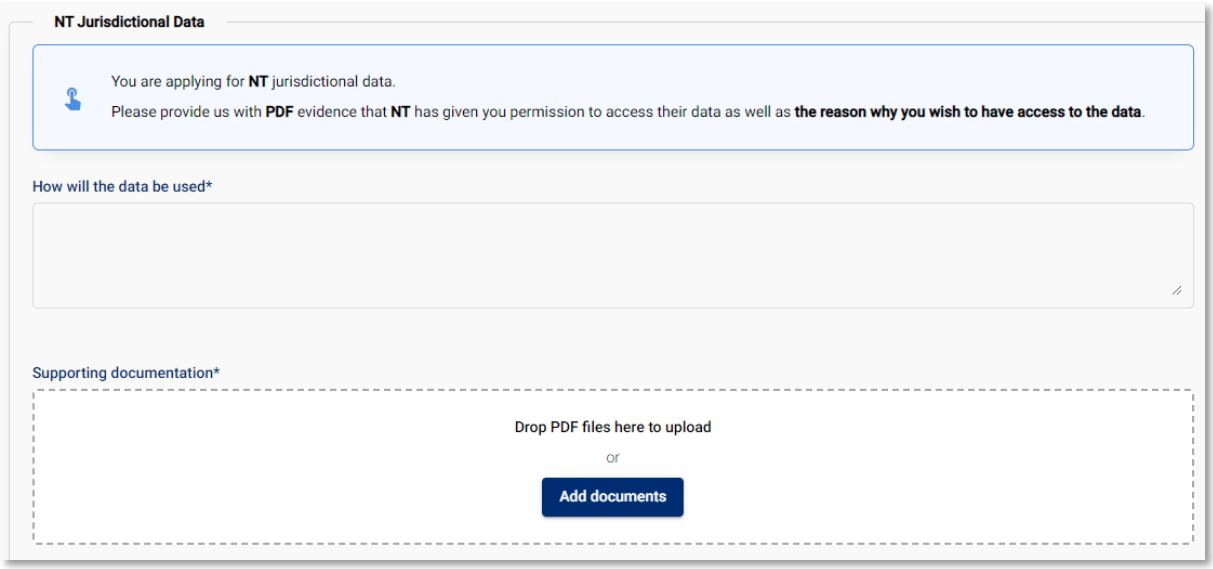
The list you are taken to shows you all of the jurisdictional data that is available in Topaz, as well as the data you have access to and the data you have applied for but which has not yet been approved. You can see the status of any application, provide further detail or revoke your application at any time by clicking on the button to the right of the State.

For example, clicking on [View Status] for ACT in the example above shows you all the details about your application. You can also add more supporting documents and change the reason you wish to gain access to the data.



However, once your application has been accepted, you can no longer perform any of those actions. Instead, all you will be able to do is revoke your data access.

Applying for data access, as mentioned above, requires that a user has the authority to get that data and evidence to that effect. When you click on the [Apply] button you are taken to a page that allows you to enter all the relevant details for the application.



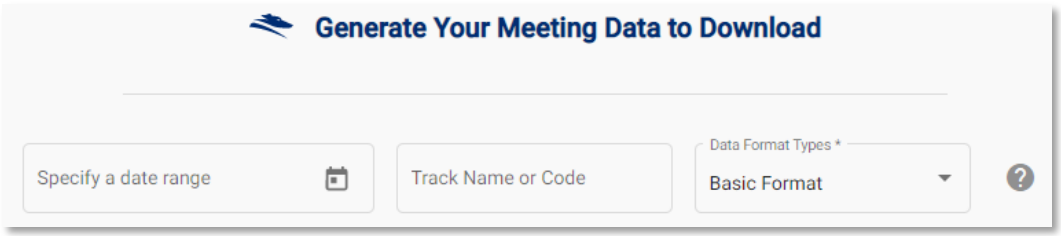
All of the data requested above is required before the user is allowed to proceed. Once all the data has been provided, the [Submit] button will become activated and when clicked, the application is placed into the GRV Topaz Staff Portal for review.

If an application is rejected by GRV, then an email will be sent to the account holders email address and the user can then log into Topaz and make the necessary changes to their application and re-submit.

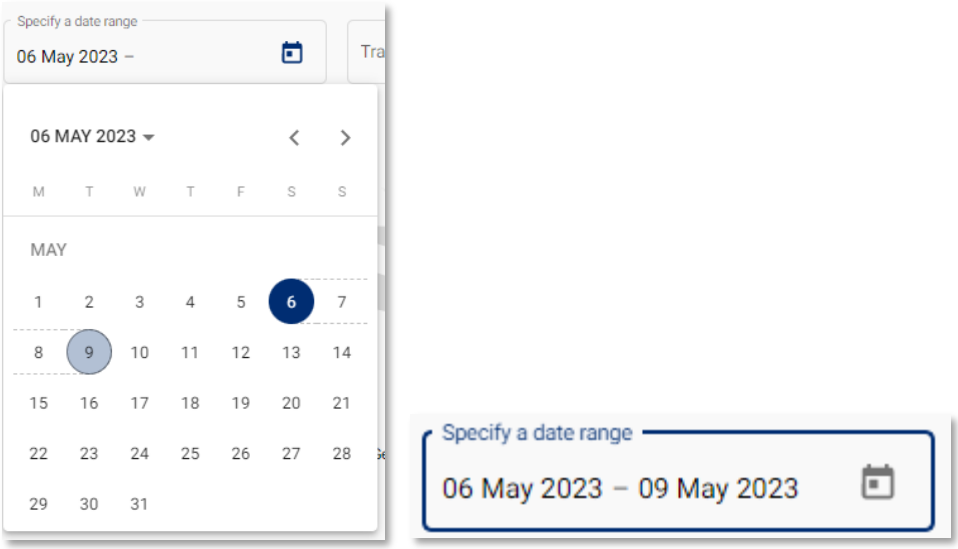
Download Data

Whilst the primary use for Topaz is programatically via the API, GRV recognises that not all users have the ability to use it or even the knowledge on what an API is. For that reason, the user portal has a download facility that users can use to download the relevant data and then enter that data into a system of their own like Excel. You do not need to have an API Key for this or any other special abilities other than to be able to read the returned data. The explanation on how to read this data is provided later on in this guide.

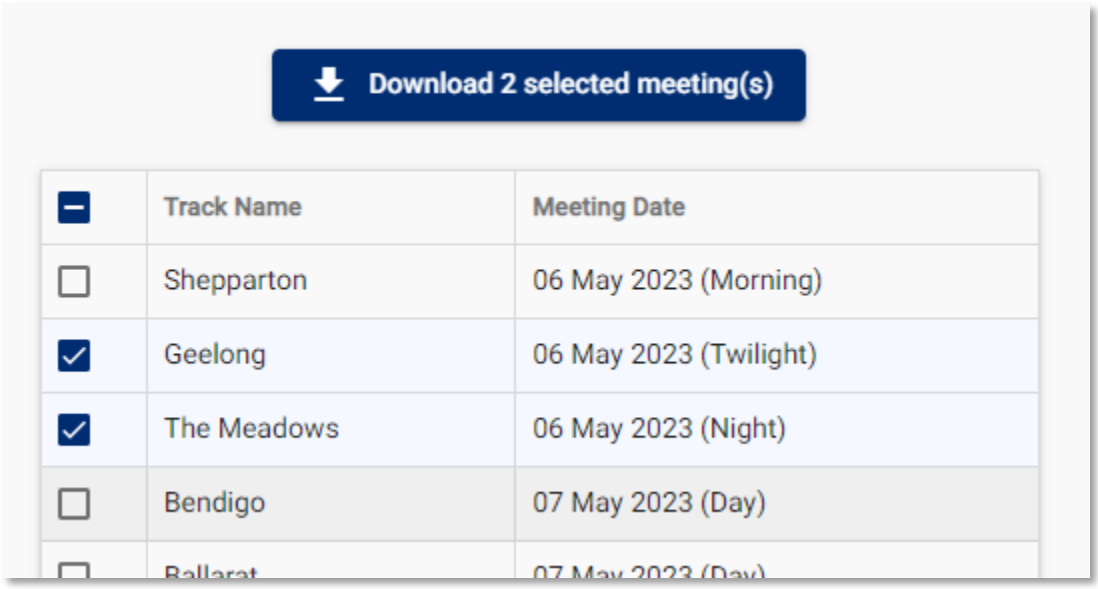
To access the download facility, click on the menu item titled [Download Data]. You are taken to a screen that will allow you to make some basic choices on which data you want.



Selecting a date range is a simple process of selecting the start date followed by the end date. When you click on the [Specify a date range] field, you will be presented with a calendar.



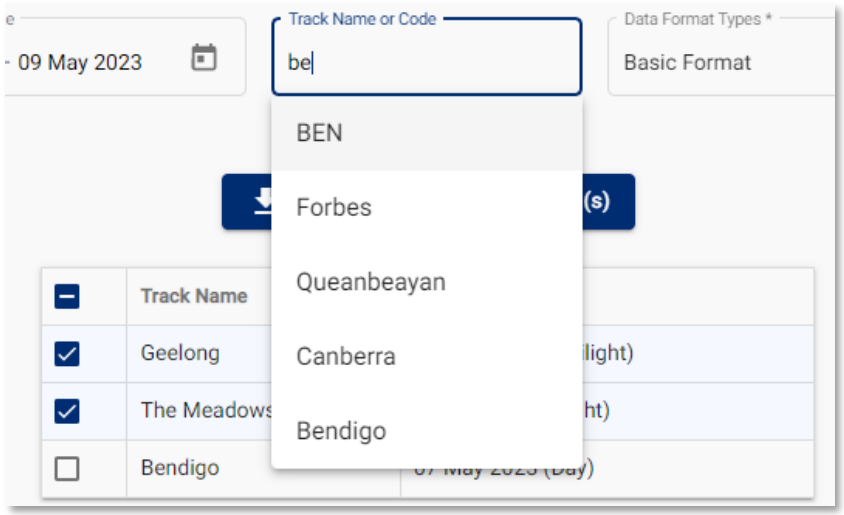
The first date you select will be the "From" date. Then simply click again to select the "To" date. In the example above, we have selected a From date of the 6th of May with a To date of the 9th of May. Once you have selected the To date, the calendar will hide and you will see a list of tracks that have data for that date range.



Important: The list of tracks you are presented with are only the data you are allowed to access. You will be given access to Victorian data as well as any other jurisdiction you have applied for and been authorised for.

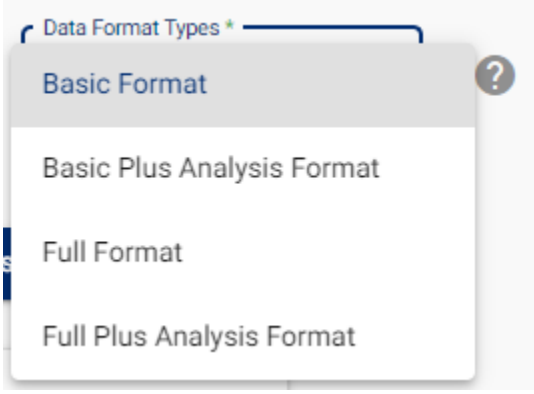
You will need to click on the check box next to each track to indicate that you want to download that data. Once you have selected all the tracks you want, you can press the [Download x selected meeting(s] button to begin the download. The more tracks you select, the longer it will take to generate and then download the data.

If you find that the list of tracks is very long, you have the ability to filter on tracks.



As you type into the [Track Name or Code] field, the list of available tracks is filtered. The selected tracks will still appear on top of the list as you filter.

Finally, you need to select the type of data you want to download. You have four format available to you.



Format Name	Description
Basic Format	This format provides just basic fields data for a meeting.
Basic Plus Analysis Format	This format provides basic fields data for a meeting plus some additional form analysis data.
Full Format	This format provides the same base format as “Basic” with the addition of data for a dog's last five starts if they are available.
Full Plus Analysis Format	This format provides the same base format as “Full” with the addition of form analysis data.

Choose the appropriate format and then download the data. A full explanation of each format is outside of the scope of this guide so the best way to determine which you need for future purposes is to select a single track, download the data in each format and take a look.

Manage Your Staff

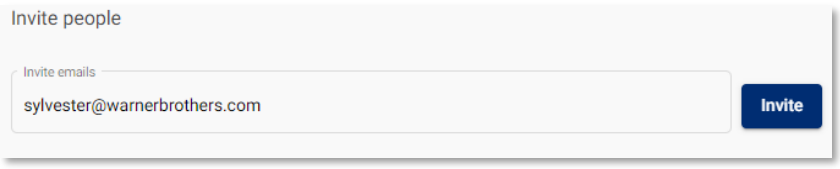
Topaz corporate accounts have the ability to add additional logins to the account. The reason we allow this is that we recognise there are organisations that will have technical staff and staff that are non-technical. We also know that staff come and go and accounts should not be inaccessible in cases where staff members either go on leave or leave the company.

Corporate users can be either an "Owner" or a "User". Owners have a few special rights like being able to create API Keys, add or remove users or generate API Keys. You can have as many users as you like with as many Owners and Users as you like but you must have at least one Owner.

One restriction on users is that the other staff members must have the same domain email address as the all the others. For example, if your email address is daffey@warnerborothers.com.au then every other staff member must have [their name]@warnerbrothers.com.au as their email address. Essentially, everything after and including the @ symbol must be the same for each staff member.

Therefore, bugsbunny@warnerbrothers.com.au, elmerfudd@warnerbrothers.com.au and foghorn@warnerbrothers.com.au are all valid staff members. However, tweety@gmail.com is not valid because Tweety does not have @warnerbrothers.com.au at the end of their email address.

Inviting people into the corporate account is done via the [Invite People] section. All you need to do is enter the email of the user you want to invite and click the [Invite] button.



An email will then be sent to that person with a link to join. Only once they have accepted the invitation will they appear in the list of users.

Technical Section

The following sections deal are technical in nature and intended for users and corporations that have some sort of ICT capability around integrating and calling API's.

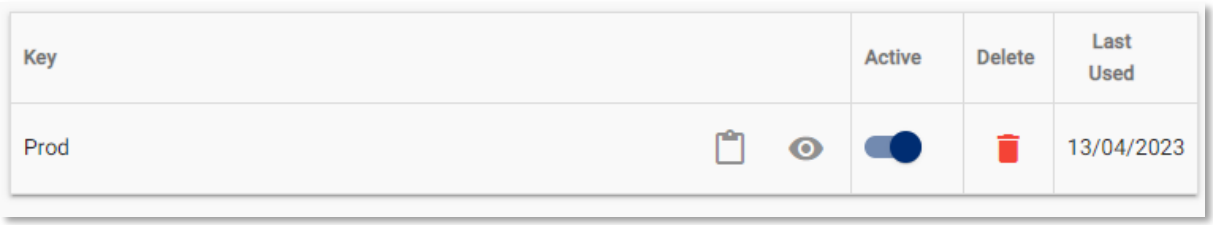
API Key Management

To call any of the API's within Topaz, you will need an API Key. That key can be generated within the portal by clicking on the [Generate a new key] button in the API Key Management page. You will be asked to give it a name before the key will be generated. This name is not used in the creation of the key, it serves only to help you organise them logically as you are able to generate more than one.

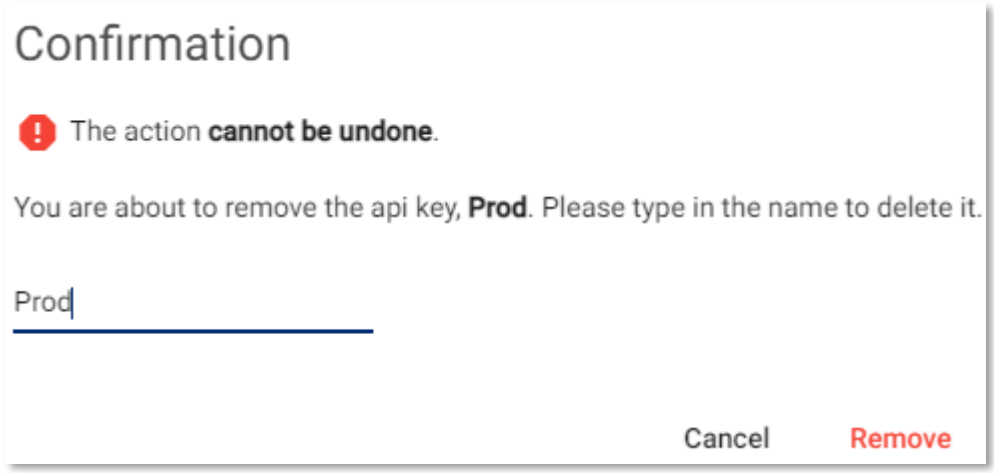
Our recommendation is that you create a key for each of your environments. So you might have a Development key, Testing and UAT as well as a Production. You may also have application specific keys. It's entirely up to you.

Within the list of keys, you have quite a lot of control over them. You can view them, copy them to the clipboard, deactivate/activate them as well as delete them. That last one should be used when you feel as though your key may have been compromised and you need to replace it.

Additionally, you can see its last used date which, if you have application level keys, can give you a little insight on your usage.



Please note however, that once a key is deleted, there is no way to recover it. It is removed permanently from all our systems and can no longer be used.



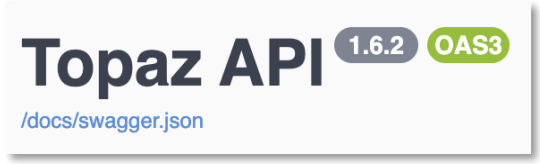
This happens immediately after you click on the "Remove" link after entering the name of the key for confirmation purposes.

Swagger

Swagger is a way to use/test an API in an environment by executing the API against either a testing database or a production one. The benefit here is that API's can be tested by anyone to ensure they perform the roles they were designed to. This would be particularly with a rules engine that governs our sport. Rules could be tested for their relevance by anyone that has access including participants.

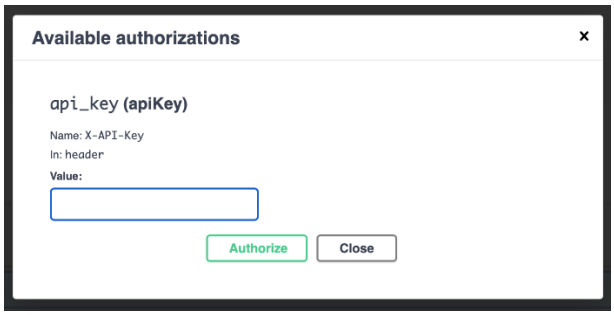
At the time of this document, the services that use swagger are Topaz and WSP Upload and have been used to good effect by our wagering service providers and the other jurisdictions.

To run any of the API's on this Swagger page you will need to be granted access to a swagger page such as the Topaz one here https://topaz.grv.org.au/docs/ and be given a key. This is a normal security measure that ensures our API's are kept safe and used only by authorised users. You can generate your own keys as described in an earlier chapter.

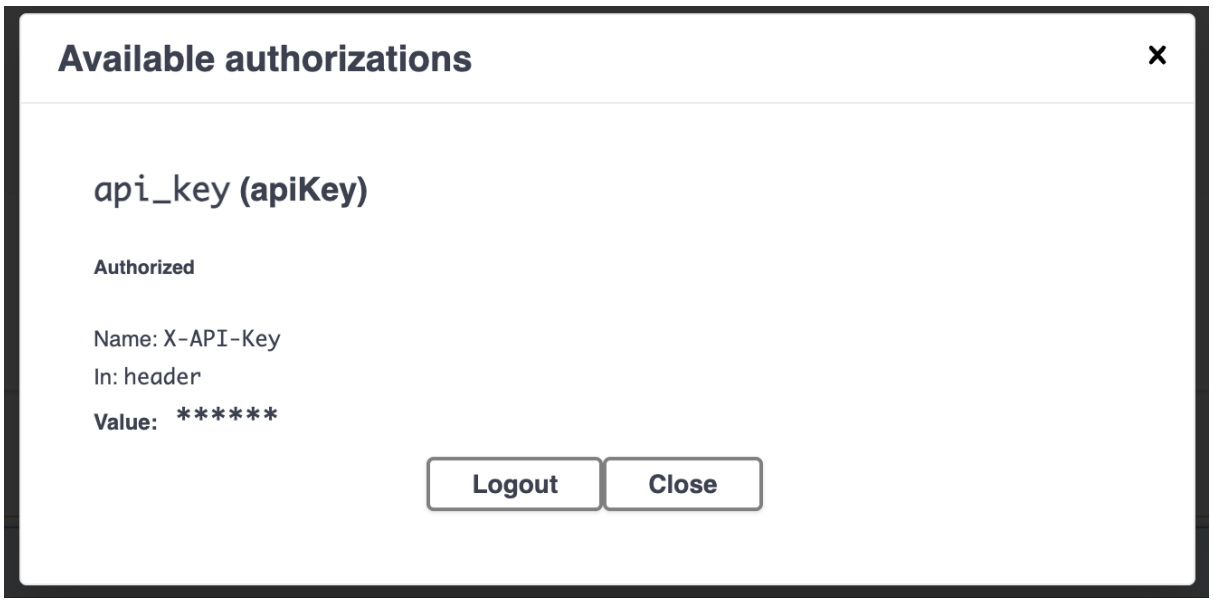


The first thing to pay attention to is the version number which tells you whether there have been any updates since you last used the Swagger pages.

Next you will need to authorise yourself as a valid user by entering your API key that was either provided to you by GRV or was generated by a system such as Topaz.



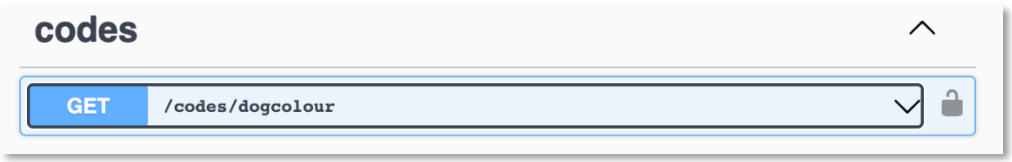
Paste your key in the provided field and press the [Authorize] button. If it is correct, then you will see that you have been "authorized" and you will see a "logout" button. If you are using a shared computer, then it is advisable to logout before leaving your desk.



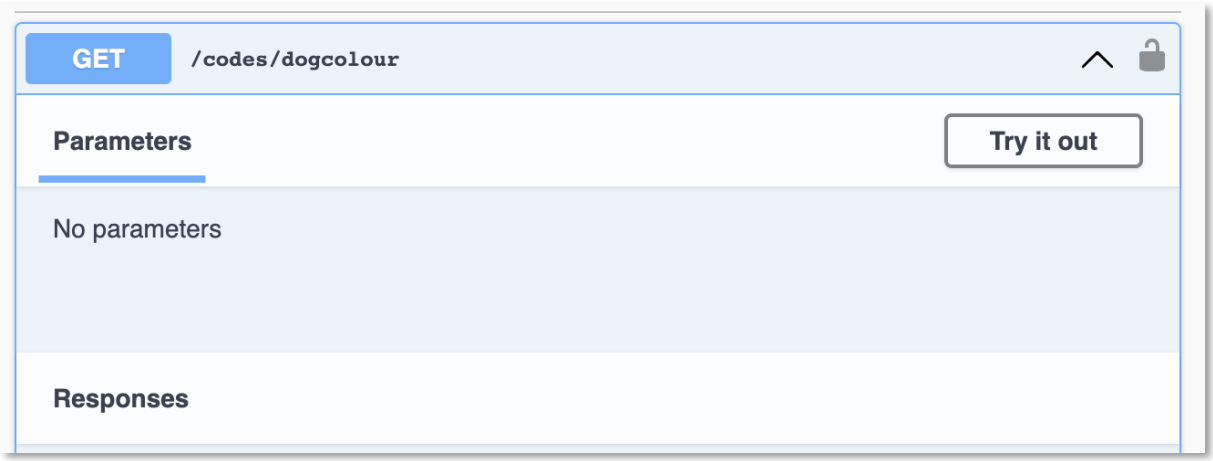
Running an API

There can be potentially hundreds of API calls that can be made from our services but we do try to segregate them into logical headings. In the case of Topaz we have headings such as Codes, Dog, FirstSplit etc to indicate what that grouping of API's does. Hopefully this helps when you are looking for the call you need.

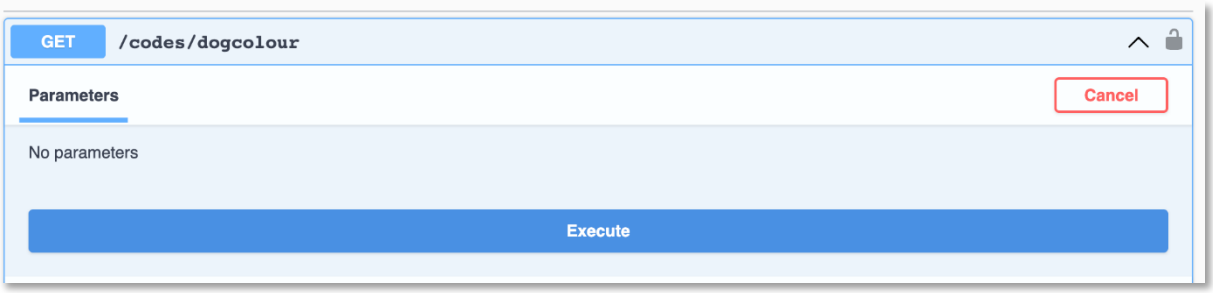
Taking a look at the top section called "codes", you can see something that has */codes/dogcolour*.



Clicking on the down arrow will open this call up and show you the option to [Try it out].

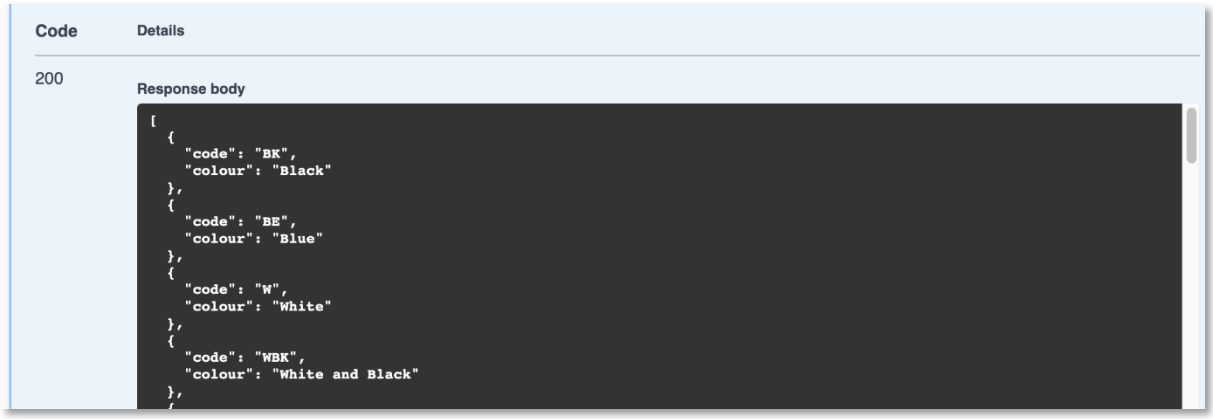


Clicking on [Try it out] will expand to show you a button labeled [Execute]. Clicking on that button will run that call.



Once run, and this may take a few seconds depending on how complex the call is, you will see a "response" section where the results of that call are displayed. Whilst this is shown in a format called JSON, care has been taken to make it as humanly readable as we could make it without compromising on functionality.

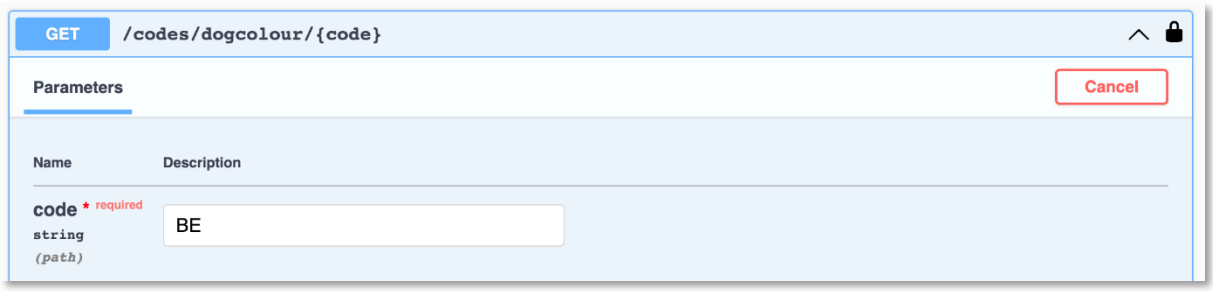
So in the below image you can see that all the dog colour codes are listed with the associated colour name.



In many cases you will need to combine more or one call to get what you need. Remember, these calls are first and foremost to be used by computer programs and so if they need to take 6 or seven steps to get what they want, this is done in les time than it can take to blink.

Above you should be able to see that one of the listed colours is blue with a code of BE. In the next call which is */codes/dogcolour/{code}* you are asked to supply the code to search for in the text field.

Type in BE and then press the execute button. You are now presented with that single colour in the response.



This is obviously a simplistic example and one that you would likely never perform. However it outlines that at times you may need to make several calls to get the data you need.

For example, if you want to obtain the IsoLynx data for a particular race you need to take several step.

Step 1: Get the track codes by executing */codes/track*


```
{
"trackId": "300",
"trackCode": "MEA",
"trackName": "The Meadows",
"owningAuthorityCode": "VIC",
"active": true
}
```


The response from this is to get every single track code that we have and you need to find the track and note it's code. For example, we would note MEA for The Meadows. If you already know the track code, then I hope it's obvious that you could have skipped this step.

Step 2: Get the meeting by executing */meeting*
The next call you need to make is to */meeting* which will list all the details about a meeting but you will need to put an entry into each of the fields labeled as "Date from", "Date to" and "TrackCode" that you got in the previous call. In our example that is MEA


```
{
"meetingId": 793731557,
"trackId": "300",
"trackCode": "MEA",
"trackName": "The Meadows",
"owningAuthorityCode": "VIC",
"meetingDate": "2023-02-18T00:00:00.000Z",
"meetingType": "Metropolitan Full Stakes",
"meetingCategory": "Metro",
"statusCode": "ResultsFinalised",
"timeSlotCode": "Night",
"nominationsCloseDateTime": "2023-02-15T04:00:00.000Z",
"scratchingsCloseDateTime": "2023-02-17T22:30:00.000Z",
"raceCount": 12,
"tabEvent": true,
"isQuali": false,
"startTime": "2023-02-18T07:37:00.000Z"
}
```


When you press the [Execute] button, what you get back, if there is a meeting that day at that track, is something like in the example above. What we want from that data is the "meetingId".

Step 3: Get the race id's by using */meeting/{meetingid}/races*
Now that we have the meeting id, which is simply a unique name for a particular meeting, we can use that to get all the races in that meeting by using */meeting/793731557/races*.
Executing this gives you a list of all the races.

You don't need to use the above, all you need to do is paste the meeting id in to the field and press the [Execute] button and you will see a list of races like below.


```
{
"raceId": 874278088,
"meetingId": 793731557,
"raceNumber": 1,
"raceTypeCode": "SH",
"raceType": "S/E Heat",
"name": "SPORTSBET FANTA BALE SUPER STAYERS HT1",
"startTime": "6:37PM",
"raceStart": "2023-02-18T07:37:00.000Z",
"distance": 730,
"isBoxDrawn": true,
"prizeMoney1": 10000,
"prizeMoney2": 3000,
"prizeMoney3": 1500,
"prizeMoney4": 750,
```

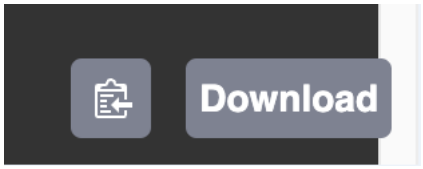

Step 4: Get the split data by using */isolynx/{raceid}/splits*
Now that we have the name of the race, or raceId, we can finally use that to get the IsoLynx data by calling */isolynx/874278088/splits*. Again, you do not need to use it like that, you just need to paste in the raceId.


```
{
"raceId": 874278088,
"splits": [
{
"averageSpeed": 60.45,
"averageToRail": 0.99,
"raceNumber": 150448,
"boxNumber": 1,
"place": 3,
"resultTime": 43.004,
"maxSpeed": 66.974,
"maxToRail": 7.86,
"minToRail": 0,
"dogName": "MASTER THE ART",
"rugNumber": 1,
"jumpSplit": {
"averageSpeed": 48.69,
"averageToRail": 5.22,
"elapsedTime": 2.15,
"maxSpeed": 52.866,
"maxToRail": 7.86,
,
"runnerSplitsFromStart50": [
{
"averageSpeed": 53.32,
"averageToRail": 2.62,
"elapsedTime": 4.01,
"maxSpeed": 58.723,
"maxToRail": 7.86,
"minToRail": 0.03,
"positionInRunning": 6,
"splitDistance": 50,
"splitSpeed": 58.723,
"deltaSpeed": 58.723,
"deltaTime": 4.01,
"splitNumber": 1,
"splitToRail": 0.23
},
```

Summing Up:
How do you know which order to call all these in?

The rule of thumb is to find the call that you want, in this case it was to obtain the IsoLynx Splits data using */IsoLynx/{Race Id}/splits*. When you expand it you will see what "parameters" it needs to be able to get you that data. In this case it was raceId. So now you work backwards until you reach the top level such as *codes/track* if you do not know the Track Code or */meeting* if you do know the Track Code.

Formatting:
To convert the JSON response to CSV use an online JSON to CSV converter such as https://www.convertcsv.com/json-to-csv.htm. In each of the responses, there are always two buttons that allow you to copy the data to the clipboard or download it to a file.



Reading JSON Data

The reading of JSON data, which is the format in which Topaz returns all data, is not all that difficult. We have taken care to name all of the elements that make up the data, in such a way that hopefully it is human readable.

The data is always bounded by an open [and a closed]. This is not arbitrary, it is a standard.

So taking a look at the JSON below, and ignoring the curly braces for now, we hope that you can see that each of the lines in the data is labelled clearly enough as to make sense. So "trackName" is clearly the name of the track. The gotchas will be with dates and times. These are always given in Zulu time and you must accommodate for that in any system that uses this data.

```
[
{
"meetingId": 793731557,
"trackId": "300",
"trackCode": "MEA",
"trackName": "The Meadows",
"owningAuthorityCode": "VIC",
"meetingDate": "2023-02-18T00:00:00.000Z",
"meetingType": "Metropolitan Full Stakes",
}
]
```

Secondly, the data may contain more open [and closed] in between the top and bottom ones. This represents data that is nested. What is meant by that is that in a meeting you will have multiple races. These races are then sometimes, depending on the API that is called, nested within the meeting.

For example, and please note that the amount of data represented below has been heavily truncated.

```
[
{
"raceId": 874278088,
"meetingId": 793731557,
"raceNumber": 1,
"raceTypeCode": "SH",
"raceType": "S/E Heat",
"name": "SPORTSBET FANTA BALE SUPER STAYERS HT1",
"runs": [
{
"trackCode": "MEA",
"track": "The Meadows",
"distance": 730,
"raceId": 874278088,
},
{
"trackCode": "MEA",
"track": "The Meadows",
"distance": 730,
"raceId": 874278088,
}
]
}
]
```

So here you can see that we have a race. We know that because of the API we called initially. But, what you also see is a label called "runs". This is a list of all the runners within that race.

Critically though, the runs label has a [directly after it. This tells you that there are "nested" items, in this case the runners, attached to the data element. And you can see that directly after the [there is a { which is the bounding bracket for a section of data which is closed by a }.

So in the above example we have the main { near the top, then we have another one at the bottom. Inside Runs we have two { } showing that there were two dogs in that race.

So to recap.

```
[ 'Opening symbol.
{ 'Opening data section
"raceId": 874278088, 'Data element
"meetingId": 793731557, 'Data element
"runs": [ 'Data element with a bounding [ indicating a nested data section. In this
case the runners.
{ 'Opening bracket for a data section
"raceId": 874278088, 'Data element
}, 'Closing bracket for the data section
{ ' New data section
"raceId": 874278088, 'Data element
}, 'Closing of the data section
] 'Closing of the nested section
} 'Closing of the main data section
] 'Closing of the entire data
```

Using this Google search gives you a lot of good resources for reading JSON type data.

"how to read json for beginners"

Or, the direct google search link.

https://www.google.com.au/search?q=how+to+read+josn+for+beginners&sxsrf=APwXEdezAaVJx90WEcGkCWfjjzoOVOoNhw%3A1683606115819&source=hp&ei=Y8pZZOe9L8Sm2roPpfWqoAM&iflsig=AOEireoAAAAAZFnYc8glTR1GHhD2WruBlw-uklw7-fNX&oq=how+to+read+JOSN+for+be&gs_lcp=Cgdnd3Mtd2l6EAMYADIHCCEQoAEQCjIHCCEQoAEQCjIKCCEQFhAeEB0QCjIKCCEQFhAeEB0QCjIKCCEQFhAeEB0QCjIKCCEQFhAeEB0QCjIKCCEQFhAeEB0QCjoHCCMQigUQJzoECCMQJzoLCAAQgAQQsQMQgwE6EQguEIAEELEDEIMBEMcBENEDOg4ILhCABBCxAxDHARDRAzoRCC4QgwEQxwEQsQMQ0QMQgAQ6CwgAEIoFELEDEIMBOgUILhCABDoRCC4QgAQQsQMQgwEQxwEQrwE6BQgAEIAEOggIABCKBRCxAzoICAAQgAQQsQM6CAguEIAEENQCOgcIABANEIAEOgYIABAeEA06CAgAEIoFEIYDOgYIABAWEB5QAFj4K2DYN2gAcAB4A4ABvASIAc4qkgEKMC4xMi45LjUtMpgBAKABAQ&sclient=gws-wiz