2 - Making API Requests

Making APIv1 Requests

API requests can be generated within the web UI by the API Request Generator, or generated programmatically in any language.

An API request looks like this: https://[your instance]/ex/api/v1/api.php?target=ipam&action=get&type=IP&mask=24

An API response is a JSON-encoded text string, and looks like this:


```
{"success":1, "message":"1 blocks found", "data":[{"id":"7539","oct1":"1","oct2":"2","oct3":"3","oct4":"0","
mask":"24","child1":null,"child2":null,"is_assigned":"0","is_swipped":"0", "is_aggregate":"1","custid":"
holding","last_updated_time":"2012-03-20 09:49:00","description":null,"parent":null,"rir":"ARIN","notes":"
2012-03-20 09:49:00","generic_code":null,"region":null,"vlan":null,"arin_net_id":null,"arin_cust_id":null,"
arin_swip_time":"0000-00-00 00:00:00","assigned_time":"2012-03-20 09:45:12"}]}
```


Instructions on decoding this return data can be found in the API endpoint documentation pages.

Using API Keys:
When using the API without pre-established authentication to ProVision, you must include both your API Key and a specially-prepared query hash parameter, like so:

> https://[your instance]/ex/api/v1/api.php?target=ipam&action=get&type=IP&mask=24&apiKey=00-TMHQV8CV2XZYABCD&hash=8jxj4IApYmgb5IZ0wBY4tFv+WilXb5JuVpjrwupyXQo=

API Keys can be generated from your ProVision instance by navigating to the Admin panel by using the gear icon in the upper right hand corner, then navigating to the API tab. The API tab will present the API authentication information in the following format:

> API Key: 00-TMHQV8CV2XZYABCD
>
> Secret Key: 6e04e5822ce10fecc8947dedxc46878

The secret key serves as an API password and is used in the creation of the API Authentication hash. The formula for creating a API query hash from an API query and a Secret Key is the following:

> Hash = Base64Encode(Sha256HMACHash (QueryString, SecretKey))

In PHP, this would be performed with the following line of code:

> $hash = *base64_encode(hash_hmac('sha256', $_SERVER['QUERY_STRING'], $secretKey, TRUE));*

⚠ **Because the hash function is computed based on the query string, you must calculate a unique hash for every API request!**

Example

Lets say you wanted to create a hash for the following API request:

> https://[your instance]/api/v1/api.php?target=ipam&action=get&type=IP&mask=24

And that your API Key and Secret Key are as follows:

> API Key: 00-TMHQV8CV2XZYABCD
>
> Secret Key: 6e04e5822ce10fecc8947dedxc46878

The first step is to append your API Key to the URL. The API Key indicates which user is executing the API query.

> https://[your instance]/api/v1/api.php?target=ipam&action=get&type=IP&mask=24&apiKey=00-TMHQV8CV2XZYABCD

The next step is to isolate the Query String from the request URL. The Query String is everything which follows the question mark. So,

> Query String: target=ipam&action=get&type=IP&mask=24&apiKey=00-TMHQV8CV2XZYABCD

The next step is to calculate the SHA256 hash of this string with your Secret Key. In PHP, this would be:

> $sha256 = hash_hmac('sha256', "target=ipam&action=get&type=IP&mask=24&apiKey=00-TMHQV8CV2XZYABCD", "48b278ec873bda4738923dbc467f8669", TRUE);

As this value has been 256-bit hashed, it will contain many unprintable characters. The solution to this is to encode it in base 64 for transport. Again, in PHP:

> $hash = *base64_encode($sha256);*

Calculating it out yields the completed hash:

> $hash = yneSFMyxPPe+3W4IOkVp50K3VStatBcRRak+2ygDUWQ=

The calculated hash can then be appended to the full API Query URL to form a completed request:

> https://[your instance]/api/v1/api.php?target=ipam&action=get&type=IP&mask=24&apiKey=00-TMHQV8CV2XZYABCD&hash=yneSFMyxPPe+3W4IOkVp50K3VStatBcRRak+2ygDUWQ=

A Note on False Positives

ProVision utilizes several possible authentication schemes of which key-based API authentication is only one. Session-based, username/password authentication is used for the majority of user interaction with the ProVision front end. Because session information is stored in browsers cookies, a browser can be authenticated to execute API commands as long as the session is active.

Unfortunately, this can lead to confusion when using a machine-based API as the user might use an authenticated browser session to test API-Key based API queries. These queries will always succeed regardless of whether the API Query Hash was calculated correctly as the system defaults to Session-based authentication when it is available.

To ensure that session-based authentication is not polluting your API-Key based testing, always use a separate browser which is not logged in to your ProVision instance to test API queries.

Other Languages

The 6Connect API can be used in just about any scripting or programming language. We have a PHP SDK that provides example code, and several useful functions for interacting with the API. Even if you don't want to use PHP, the samples will help you create code in other languages

A Note on False Positives

ProVision utilizes several possible authentication schemes of which key-based API authentication is only one. Session-based, username/password authentication is used for the majority of user interaction with the ProVision front end. Because session information is stored in browsers cookies, a browser can be authenticated to execute API commands as long as the session is active.

Unfortunately, this can lead to confusion when using a machine-based API as the user might use an authenticated browser session to test API-Key based API queries. These queries will always succeed regardless of whether the API Query Hash was calculated correctly as the system defaults to Session-based authentication when it is available.

To ensure that session-based authentication is not polluting your API-Key based testing, always use a separate browser which is not logged in to your ProVision instance to test API queries.

Other Languages

The 6Connect API can be used in just about any scripting or programming language. We have a PHP SDK that provides example code, and several useful functions for interacting with the API. Even if you don't want to use PHP, the samples will help you create code in other languages.