

APIv2

APIv2



API version 2 2.0.0 OAS3

Explore our API families.

[IPAM API](#)

Includes actions for LIRs, IP aggregate and block management, VLAN, IP Rules, and SWIP.

[Resource API](#)

Includes actions for managing the [Resource System](#).

The resource API provides CRUD endpoints for resources, resource attributes, resource attachments and resource backups.

[Resource View API](#)

The resource view API provides CRUD endpoints for resource views.

[DNS API](#)

DNS API allows you to manage DNS Zones, Records, Servers, Groups and ACLS.

[Peering API](#)

Peering API allows you to manage Routers and BGB Sessions.

[Contacts API](#)

Contacts API allows you to manage contacts.

[Users API](#)

Includes actions for Users, permissions and actions.

[Usergroups API](#)

Includes actions for Groups, permissions and actions

[Scheduler API](#)

The API Allows you to easily schedule tasks.

[API Composer Platform](#)

API Composer Platform (ACP) is an additional module in to help automate frequently used combinations of calls.

[SSH API](#)

Includes actions for SSH Credentials and SSH Routes

[Terms of service](#)

[Send email to our dev team](#)

[License information](#)

- APIv2
 - API v2 Overview
 - APIv2 Access Options
 - APIv2 - Using Swagger
 - Accessing Swagger
 - Viewing APIv2 Information
 - Testing/Executing Endpoints in Swagger
 - APIv2 - Nested Queries
 - APIv2 - Using cURL
 - cURL syntax template for APIv2:
 - cURL Examples
 - Additional Information

[API v2 Overview](#)

APIv2 is ProVision's currently supported RESTful API version. APIv2 improvements include:

- HTTPS Basic Authentication
- Use of HTTP Methods (GET, PUT, POST, etc.)
- Supports JSON payloads
- Additional endpoints and functionality

APIv2 Access Options

To test or execute APIv2 queries, you may:

1. Use a browser extension / desktop REST client, such as [Postman](#)
 - a. Postman is the current industry standard: Go to <https://www.getpostman.com/> to install, and visit the [Postman Learning Center](#) for user documentation, training videos, and support help.
2. Access the APIv2 Swagger documentation from your instance (*instance*/dev/swagger), which provides the ability to test inputs and responses using your instance data.
 - a. Continue to the section below: "APIv2 - Using Swagger" for more details, or see the [Developer Tools](#) page for a broader overview.
3. Use CURL in the command line to authenticate and execute APIv2 endpoints. See cURL documentation at <https://curl.haxx.se/>.
 - a. Continue to the section below: "APIv2 - Using cURL" for more details, examples, and tips.

APIv2 - Using Swagger

Accessing Swagger

Public APIv2 documentation is located at <https://cloud.6connect.com/APIv2/>.

APIv2 documentation includes:

- [IPAM API](#)

Includes actions for LIRs, IP aggregate and block management, VLAN, IP Rules, and SWIP.
- [Resource API](#)

Includes actions for managing the [Resource System](#).
The resource API provides CRUD endpoints for resources, resource attributes, resource attachments and resource backups.
- [Resource View API](#)

Provides CRUD endpoints for resource views.
- [DNS API](#)

The DNS API allows you to manage DNS Zones, Records, Servers, Groups and ACLS.
- [Peering API](#)

Allows you to manage routers and BGP sessions.
- [Contacts API](#)

Allows you to manage contacts.
- [Users API](#)

Includes actions for Users, permissions and actions.
- [Usergroups API](#)

Includes actions for Groups, permissions and actions
- [Scheduler API](#)

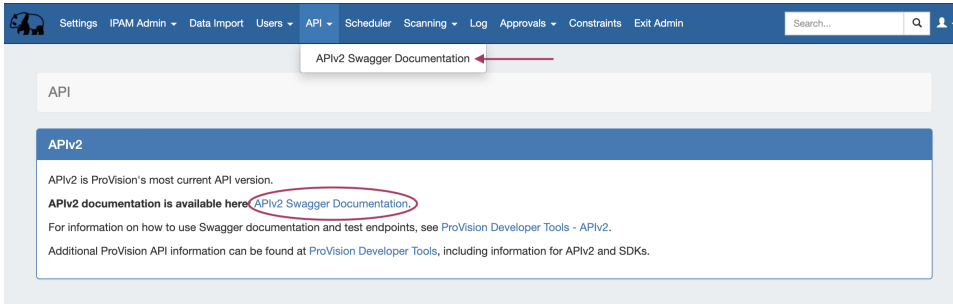
The API Allows you to easily schedule tasks.
- [API Composer Platform](#)

API Composer Platform (ACP) is an additional module to help automate frequently used combinations of calls.
- [SSH API](#)

Includes actions for SSH credentials and SHH Routes.

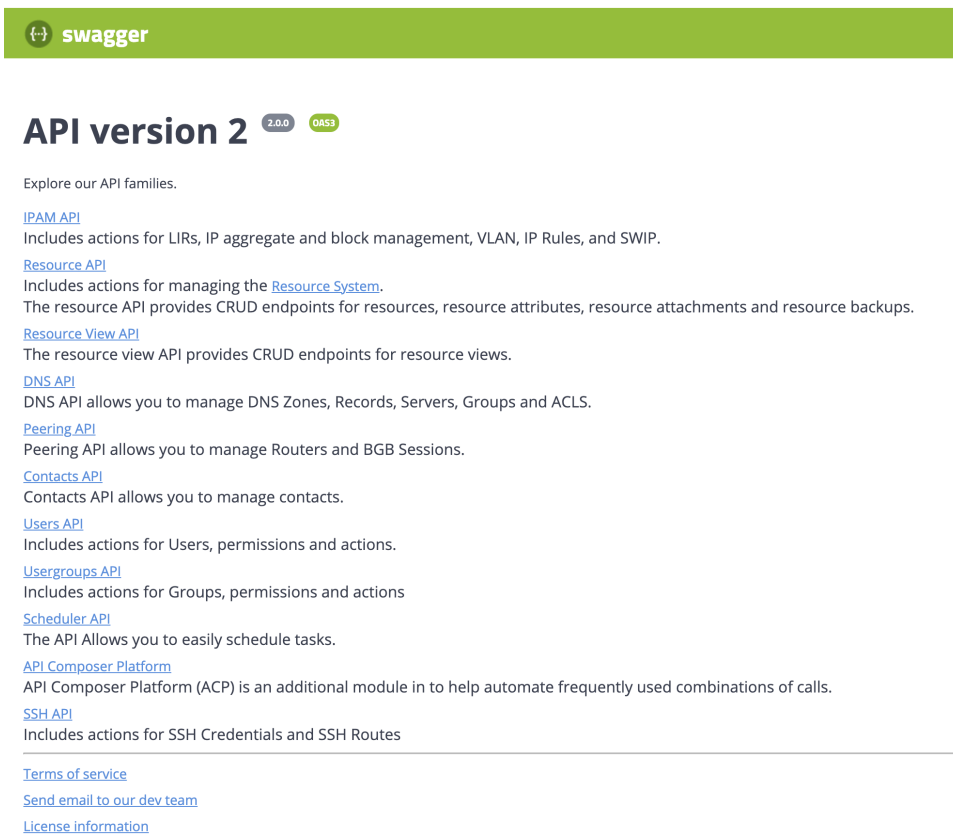
Existing customers may also access APIv2 Documentation from your instance (user must have Admin permissions):

1. Log into your ProVision instance.
2. Go to the Admin area of ProVision and click on the [API](#) Tab.
3. Under the API dropdown menu, click the Swagger link provided. You may also click on the "APIv2 Swagger Documentation" link provided under the APIv2 section of the page.



Viewing APIv2 Information

1. On the API Swagger home page, click on the name link for the API family that you wish to browse (IPAM, Resource, DNS, etc).



2. Once on an API Family page, verify that the displayed server name is correct for your instance/local server.

In most situations, only one ProVision instance/server will be displayed, with authentication already provided from your ProVision login. If your ProVision session has ended, or the server changed, you may need to re-provide ProVision credentials by clicking the "Authorize" button.

Authorize

Server

https://

/api/v2

3. Scroll further down the page and begin reviewing available APIv2 calls and details. Clicking on any call will expand it to view parameter details - you can even test call responses (using your instance data) by clicking "Try it Out"!

The detail information includes a description, parameter list (required parameters are marked with a *), and response information

default

GET /ipam/lirs GET Lirs

POST /ipam/lirs Create LIR

GET /ipam/lirs/{id} Retrieve LIR

Returns information on a single LIR.

Parameters

Name

Description

id * required

string (path)

ID of the LIR

Responses

Code	Description	Links
200	successful operation	No links
400	Bad Request	No links
401	Invalid credentials	No links

4. Some calls that involve a JSON request body payload (PUT, PATCH, etc) will display "Example Value" and "Model" information under a "Request Body" section - additional parameter descriptions may be displayed under "Model" Information.

Clicking on "Example Value" will show an example of a JSON request body for that call.

Example Value

Model

```
{
  "name": "TestLIR",
  "rir": "ARIN",
  "asn": 20320,
  "parent_id": 0,
  "entities": [
    "string"
  ],
  "type": "entry",
  "section": "string",
  "custom_id": 0
}
```

Clicking on "Model" will display details and descriptions of the request body parameters, if available.

```
Example Value Model

put_lir {
  name: string
  example: TestLIR
  Name of the LIR resource

  rir: string
  example: ARIN
  The RIR for the LIR. Accepted values are "ARIN", "RIPE", "LACNIC", "AFNIC", "APNIC", and "1918".

  asn: integer($int64)
  example: 30002
  The ASN (Autonomous System Number) for the LIR

  parent_id: integer($int64)
  entites: > [...]
  type: string
  example: entry
  Type of resource - Updating a LIR will always be "entry"

  section: string
  Section of the resource object

  custom_id: integer($int64)
}
```

5. Additional "Model" examples are available at the bottom of the page with additional descriptive information.

At the bottom of the page, click on "Models".

```
GET /ipam/vlans/{id} Retrieve VLAN
GET /ipam/{size}/sizes GET IPAM Sizes

Models
```

Then, click on the "Model" you wish to view. Some models may contain additional information that you can expand to view, such as valid values for a parameter. In the example below, the circled "array" will display valid RIR values.

```
Models

retrieve_vlan_output > [...]
get_netblock_resource_hierarchy_output > [...]

oneOf_direct_assign_1 {
  resource_id: integer($int64)
  example: 1234
  The ID of the resource the block is assigned to

  generic_code: integer($int64)
  example: Code X
  User-defined block code as defined in Admin-IPAM settings: Generic Code Per Block Name

  lir_id: integer($int64)
  example: 101
  The numeric ID of an LIR resource the block should be linked to

  rir: string
  Regional Internet registry
  Enum:
  -> Array [ 8 ]
  integer($int64)
  example: 2
  The numeric ID of a region

  tags: string
  example: customer,vpn
  Comma-separated list of tags to filter by. If used in conjunction with 'search', performs the search operation and then filters results by the provided tag. Use with tagMode to specify filter approach.

  tags_mode: string
  Denotes how the "tags" parameter is handled:
}
```

Testing/Executing Endpoints in Swagger

You may execute endpoints in Swagger by using the "Try it out" button for any call.

1. Navigate to the call that you want to try out.
2. Expand the call to view its details, then click the "Try it out" button.

```
GET /ipam/lirs/{id} Retrieve LIR

Returns information on a single LIR.

Parameters

Name Description
id * required ID of the LIR
string (path)
```

3. Input the desired parameters to test, and click "Execute".

GET /ipam/lirs/{id} Retrieve LIR

Returns information on a single LIR.

Parameters

Name	Description
id * required string (path)	ID of the LIR

1234

Execute

Responses

If the call is a method that uses a JSON request body, you will have the option to edit the body text in the "Example Value" box - when done, click "Execute".

Example Value Model

```
{
  "name": "TestLIR",
  "id": "1234",
  "slug": "abc-2",
  "parent_id": 0,
  "entities": [
    {
      "type": "entry",
      "section": "string",
      "custom_id": 0
    }
  ]
}
```

Execute

- The example response will display under "Responses" after being executed. The "Response" section also includes the cURL command, Request URL, and Response Headers.

Responses

Curl

```
curl -X GET "https://api/v2/ipam/lirs/14501" -H "accept: */*"
```

Request URL

```
https://api/v2/ipam/lirs/14501
```

Server response

Code	Details
200	<p>Response body</p> <pre>{ "id": "14501", "name": "abc", "slug": "abc-2", "parent_id": 0, "entities": [{ "type": "entry", "section": "string", "custom_id": 0 }] }</pre> <p>Response headers</p> <pre>access-control-allow-headers: GET, PUT, POST, PATCH, OPTIONS, DELETE access-control-allow-methods: GET, PUT, POST, PATCH, OPTIONS, DELETE access-control-allow-origin: * access-control-expose-headers: X-Total-Count, Location cache-control: no-store, no-cache, must-revalidate connection: Keep-Alive content-encoding: gzip content-length: 135 content-type: application/json; charset=utf-8 date: Fri, 22 Mar 2019 02:25:42 GMT expires: Thu, 19 Nov 1981 08:52:00 GMT keep-alive: timeout=5, max=100 pragma: no-cache server: Apache/2.4.18 (Ubuntu) vary: Accept-Encoding x-timestamp: valid</pre>

APIv2 - Nested Queries

You can use nesting to write more complicated API calls to handle multiple conditions. Here, we'll illustrate a query with an OR / AND condition, such as to find resource entries with (Section = 76 OR Section = 77) AND managed = Yes:

```
{ "type": "entry", "attrs": [{ "key": "_custom_id", "value": "101", "rule": "AND" }, { "attr_rule": "AND", "attr_nested": [ { "attr_key": "_section", "attr_value": "78", "attr_rule": "OR" }, { "attr_key": "_section", "attr_value": "74", "attr_rule": "OR" } ] } ] }
```

You should be able to adapt that to point at whatever attributes you like.

Here it is in CURL format:

```
curl -X POST "https://$HOSTNAME/instance/api/v2/resources/query" -H "accept: application/json" -H "Content-Type: application/json" -d '{"type":"entry","attrs":[{"key":"_custom_id","value":"101","rule":"AND"}, {"attr_rule":"AND", "attr_nested":[{"attr_key":"_section","attr_value":"78", "attr_rule":"OR"}, {"attr_key":"_section","attr_value":"74", "attr_rule":"OR"}]}]}'
```

APIv2 - Using cURL

One option to execute an APIv2 call is to use CURL from the Command Line.

If you do not already have cURL installed, you may download it and access its documentation at <https://curl.haxx.se/>.

cURL syntax template for APIv2:

When using cURL, the general layout of the cURL command will be as follows:

1. Invoke cURL:

```
a. curl -X PATCH "https://$HOSTNAME/instance/api/v2/config" -H "Content-Type:application/json;charset=utf-8" -u testing@example.com:password -d '{"auto_merge_limits":{"10"}}'
```

2. Use the Request Flag (-X):

```
a. curl -X PATCH "https://$HOSTNAME/instance/api/v2/config" -H "Content-Type:application/json;charset=utf-8" -u testing@example.com:password -d '{"auto_merge_limits":{"10"}}'
```

b. This specifies a custom request method to use when communicating with the HTTP server. The specified request method will be used instead of the method otherwise used (which defaults to GET)

3. Specify the HTTP Action:

```
a. curl -X PATCH "https://$HOSTNAME/instance/api/v2/config" -H "Content-Type:application/json;charset=utf-8" -u testing@example.com:password -d '{"auto_merge_limits":{"10"}}'
```

4. In quotes, specify the URL of the APIv2 endpoint you are executing. This should be similar to: "https://\$HOSTNAME/instance/api/v2/[endpoint family-action]"

```
a. curl -X PATCH "https://$HOSTNAME/instance/api/v2/config" -H "Content-Type:application/json;charset=utf-8" -u testing@example.com:password -d '{"auto_merge_limits":{"10"}}'
```

5. Use the Header Flag (-H):

```
a. curl -X PATCH "https://$HOSTNAME/instance/api/v2/config" -H "Content-Type:application/json;charset=utf-8" -u testing@example.com:password -d '{"auto_merge_limits":{"10"}}'
```

b. Denotes an extra header to include in the request when sending HTTP to a server. You may specify any number of extra headers. Note that if you should add a custom header that has the same name as one of the internal ones curl would use, your externally set header will be used instead of the internal one. This allows you to make even trickier stuff than curl would normally do.

6. Add endpoint-specific data, such as specifying a JSON payload or attribute setting. This may involve one or more additional flags or information sets, depending on the endpoint:

```
a. curl -X PATCH "https://$HOSTNAME/instance/api/v2/config" -H "Content-Type:application/json;charset=utf-8" -u testing@example.com:password -d '{"auto_merge_limits":{"10"}}'
```

Tip!

Swagger displays cURL commands and request URLs in the execution response!

Use the "Try it Now" feature from your instance's Swagger page (Accessed from Admin API Tab APIv2 Swagger Documentation) for the endpoint/attribute changes you wish to make, and view the cURL command for that change. Copy the command text, and use it as a template for your next cURL execution of the command!

Responses

Curl

```
curl -X GET "https://[redacted]/api/v2/resources" -H "accept: application/json"
```

Request URL

```
https://[redacted]/api/v2/resources
```

Server response

Code	Details
------	---------

200

For help using Swagger to test endpoints, see "Testing Endpoints in Swagger" in this APIv2 documentation page.

cURL Examples

Example: cURL APIv2 Authentication

This example authenticates a user, so that you may perform APIv2 commands as that user.

```
curl -X GET "https://$HOSTNAME/instance/api/v2/config" -H "accept: */*" -u testing@example.com
```

This will ask for a password. To hard-code it with a password, add it to the end like so:

```
curl -X GET "https://$HOSTNAME/instance/api/v2/config" -H "accept: */*" -u testing@example.com:password
```

APIv2 commands are executed as the user provided, so their permissions must be set appropriately. The Swagger "execute" feature produces CURL strings that can be used to test specific API commands.

Example: cURL APIv2 POST Command

This example illustrates how to provide POST/PATCH data to an APIv2 command via cURL. This command updates the automerge configuration setting:

```
curl -X PATCH "https://$HOSTNAME/instance/api/v2/config" -H "Content-Type:application/json;charset=utf-8" -u testing@example.com:password -d "{\"auto_merge_limits\":\"10\"}"
```

Review the general CURL documentation at <https://curl.haxx.se/> can offer greater insight into what other flags can be used.

Example: cURL / APIv2 Nested Query

You can use nesting to write more complicated API calls to handle multiple conditions. Here, we'll illustrate a query with an OR / AND condition, such as to find resource entries with (Section = 76 OR Section = 77) AND managed = Yes:

```
{ "type": "entry", "attrs": [{ "key": "_custom_id", "value": "101", "rule": "AND", { "attr_rule": "AND", "attr_nested": [{ "attr_key": "_section", "attr_value": "78", "attr_rule": "OR", { "attr_key": "_section", "attr_value": "74", "attr_rule": "OR" } } ] } ] }
```

The items in bold red are the necessary components to achieve the AND/OR condition in a nested format, though the attributes used may be changed depending on your needs.

The same query in cURL is given below:


```
curl -X POST "https://$HOSTNAME/instance/api/v2/resources/query" -H "accept: application/json" -H "Content-Type: application/json" -d '{"type":"entry","attrs":[{"key":"_custom_id","value":"101","rule":"AND"}, {"attr_rule":"AND", "attr_nested":[{"attr_key":"_section","attr_value":"78","attr_rule":"OR"}, {"attr_key":"_section","attr_value":"74","attr_rule":"OR"}]}]}'
```

Example: cURL APIv2 Attach Zone to a DNS Group

To attach a newly created DNS Zone to a DNS Group using APIv2, you need to make a Resource Link with the relation type "dnsZoneView":

```
curl -X POST "https://$HOSTNAME/instance/api/v2/resources/274/links" -H "accept: application/json" -H "Content-Type: application/json" -d '{"resource_id":"15648","relation":"dnsZoneView","data":{}}'
```

Where resource #274 is the "Default Group" on the instance (your instance may vary), and Resource #15648 is the resource ID for the new zone. The two are now linked together.

To attach a DNS server to a DNS Group you use the same process but the relation is "dnsViewServer".

To attach a zone to a server directly the relation is either "dnsZoneMaster" or "dnsZoneSlave" depending on what you're doing.

Additional Information

See the following areas for additional information:

- [Developer Tools](#)
- Public APIv2 Swagger documentation: <https://cloud.6connect.com/APIv2/>.