

[Paragon PathFinder API \(api-ref-pathfinder-v2.html\)](#)

[Paragon PathFinder API Developer Guide \(api-ref-pathfinder-devguide.html\)](#)

Working with Pathfinder RESTful APIs

Use the Juniper Pathfinder APIs to monitor and control your TE-LSPs.

The current API definitions can be found on the Pathfinder API v2 ([api-ref-pathfinder-v2.html](#)) page.

You can use the APIs and extensions after you authenticate through the authentication API.

The Pathfinder RESTful APIs are designed to enable access over HTTP to most of the same data and analytics that are available to you from both the Pathfinder GUI and the Pathfinder CLI.

Many Pathfinder users have little or no prior experience developing programs or scripts that make use of RESTful APIs. For this reason, we are providing a brief introduction designed to help you with some basics about REST and what it means to access data and functionality using common conventions associated with REST. In addition, we are providing a few simple code examples that illustrate common uses of specific APIs.

REST (REpresentational State Transfer) is a set of useful conventions and principals about transfer of information over the World Wide Web.

Many Web services are now using the principals of REST in their design.

When you type a URL into your browser, like `http://example.net`, your browser software creates an HTTP header that identifies:

- a desired action: GET ("get me this resource").
- a target machine (`www.domain-name.com`).

When you update your credit card information to an online account, your browser software creates an HTTP header with a payload that contains your form data;

The HTTP header identifies:

- a desired action: POST ("here's some update info").
- a target machine (`www.domain-name.com`).

The payload contains:

- form data to POST to a resource on the target machine.

The HTTP verbs (also known as methods) that are commonly used in RESTful requests are POST, GET, PUT, and DELETE. These are known as CRUD, for create, read, update, and delete operations. Additional HTTP methods exist, but are used less frequently, including PATCH, OPTIONS and HEAD, among others, but these are not used in the Pathfinder APIs

CRUD

HTTP Verb	CRUD	Entire Collection (e.g. /customers)	Specific Item (e.g. /customers/{id})
POST	Create	201 (Created), 'Location' header with link to /customers/{id} containing new ID.	404 (Not Found), 409 (Conflict) if resource already exists..
GET	Read	200 (OK), list of customers. Use pagination, sorting and filtering to navigate big lists.	200 (OK), single customer. 404 (Not Found), if ID not found or invalid.
PUT	Update/Replace	404 (Not Found), unless you want to update/replace every resource in the entire collection.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
DELETE	Delete	404 (Not Found), unless you want to delete the whole collection—not often desirable.	200 (OK). 404 (Not Found), if ID not found or invalid.

RESTful requests to Pathfinder will often follow patterns similar to the following request, which retrieves a full topology:

The data in a response message body can range from a single entry to thousands of lines, depending on the request. The following example illustrates the kind of data you might see returned in response to a topology request:

```
[{"node":{"topologyType":"node","topologyIndex":0,"name":"0840.0000.0105.02","nodeIndex":6,"AutonomousSystem":{"asNumber":84}}, {"topologyType":"node","topologyIndex":0,"name":"0840.0000.0105.03","nodeIndex":7,"AutonomousSystem":{"asNumber":84}}, {"topologyType":"node","topologyIndex":0,"name":"0840.0000.0106.02","nodeIndex":9,"AutonomousSystem":{"asNumber":84}}, {"topologyType":"node","topologyIndex":0,"name":"84.0.0.1","nodeIndex":1,"AutonomousSystem":{"asNumber":84}}, {"topologyType":"node","topologyIndex":0,"name":"84.0.0.102","nodeIndex":2,"AutonomousSystem":{"asNumber":84}}, {"topologyType":"node","topologyIndex":0,"name":"84.0.0.103","nodeIndex":3,"AutonomousSystem":{"asNumber":84}}, {"topologyType":"node","topologyIndex":0,"name":"84.0.0.104","nodeIndex":4,"AutonomousSystem":{"asNumber":84}}, {"topologyType":"node","topologyIndex":0,"name":"84.0.0.105","nodeIndex":5,"AutonomousSystem":{"asNumber":84}}, {"topologyType":"node","topologyIndex":0,"name":"84.0.0.106","nodeIndex":8,"AutonomousSystem":{"asNumber":84}}, {"topologyType":"node","topologyIndex":0,"name":"84.0.0.107","nodeIndex":10,"AutonomousSystem":{"asNumber":84}}, {"links":[{"topologyType":"link","topologyIndex":0,"name":"L111.84.111.84.2","linkIndex":12,"endZ":2}, {"topologyType":"interface","ipv4Address":["topologyType":"IPv4","address":"111.84.2"], "protocols":["RSPV": {"bandwidth":["IGP"], "bandwidth":["IGP","ETmetric":10,"ECcolor":["srts"],"unreservedBw": [1000000000,1000000000,1000000000,1000000000,1000000000,1000000000,1000000000,1000000000]}], "node": ["topologyType":"node","name":"84.0.0.105"]}], {"topologyType":"link","topologyIndex":0,"name":"L84.103.107.1_84.103.107.2","linkIndex":4,"endA":4}, {"topologyType":"interface","ipv4Address":["topologyType":"IPv4","address":"84.103.107.1"], "protocols":["RSPV": {"bandwidth":["IGP"], "bandwidth":["IGP","ETmetric":10,"ECcolor":["srts"],"unreservedBw":
```

