**Q&A**

**Q) When can a request refer to a nested resource for filtering or field selection?**

A general rule which usually works is this: if a field is **NOT** a URL by default, then you **CANNOT** “dive into it” with query parameters. For instance, if you GET https://www.cradlepointecm.com/api/v2/routers/, the “product” field is a URL:

"product": "https://www.cradlepointecm.com/api/v2/products/2"

By way of comparison, if you GET https://www.cradlepointecm.com/api/v2/alerts/, the “info” field is not a URL, it's a literal value (which happens to be a nested dict):

"info": { ... stuff ... }

So filters and field selection cannot “dive into” the internals of info.

**Q) What is the best way to get API results into Excel?**

The best way is to page the results in as JSON and convert them using a script.

**Q) How can I use the API to create an uptime report?**

If by “uptime” you mean “time the device has been connected to NCM,” then this is very close:

https://www.cradlepointecm.com/api/v2/routers/?state=online&fields=state\_updated\_at

This shows all the devices currently online and the time at which they came online. Again (with a script) one can subtract that time from the current time to get uptime. You can do the same for state=offline to see when devices were detected as offline by NCM.

As an alternative, (for series 3 products only), each net\_device reports an uptime:

https://www.cradlepointecm.com/api/v2/net\_devices/?connection\_state=connected&fields=uptime,name

**Filtering**

**Filter Syntax:**

**Multiple filters can be provided. Results will match all of the filters.**

Generic Filters

|  |  |
| --- | --- |
| fieldname=value | Exact match |
| fieldname\_\_in=[value\_1, value\_2,..., value\_n] | Match any from a list |

**Examples:**

Return all devices in groups 5, 6 or 7 that are offline:

GET https://www.cradlepointecm.com/api/v2/routers/?group\_\_in=5,6,7&state=offline

**Sorting**

**Partial Results:**

A partial result is when only a subset of the available fields are asked for in a request. The syntax.

|  |  |
| --- | --- |
| fields=field\_1,field\_2,...,field\_n | Return only the fields listed |

**Examples:**

Return only the name, state, group and mac for all devices:

GET https://www.cradlepointecm.com/api/v2/routers/?fields=name,state,group,mac

**Expands**

Expanding is when related objects are asked for in request. Only a subset of related objects allow expands. The syntax.

|  |  |
| --- | --- |
| expands=field\_1,field\_2,...,field\_n | Return related data of the fields listed |

**Examples:**

Return all devices, as well as the related objects account and group:

GET https://www.cradlepointecm.com/api/v2/routers/?expand=account,group

**Paging**

When getting a list you can request a slice with the paging parameters: 'offset' and 'limit'.

GET https://www.cradlepointecm.com/api/v2/routers/?offset=count&limit=count

Any time a list is returned the values used for offset and limit are returned in the meta portion of the reply. Note that most resources have a max limit of 500. To page the whole list you need to increment the offset. So to get the next 50 devices:

**Examples:**

import requests

headers = {

'X-CP-API-ID': '…',

'X-CP-API-KEY': '…',

'X-ECM-API-ID': '…',

'X-ECM-API-KEY': '…',

'Content-Type': 'application/json',

'Accept': '\*/\*'

}

result = []

#Get next 50 devices

url = 'https://www.cradlepointecm.com/api/v2/routers/?offset=50&limit=50'

#loop to get all the data

while url != None:

req = requests.get(url, headers=headers)

alert\_data = req.json()

result.append(alert\_data)

meta = alert\_data['meta']

url = meta['next']

**Glossary**

* admin: a user named as admin by an account, and who gains certain automatic privileges thereby.
* authentication: the act of verifying that a request is really coming from the user it claims to be coming from.
* authorization: the act of checking that a request's user has permission to do what the request is trying to do.
* CRUD: acronym for create, read, update and delete; the four basic operations on a data resource.
* DTD: document type definition; a structured document describing the format of a class of structured documents. A DTD specifies data types, defaults and hierarchical structure.