PeeringDB

PeeringDB Workshop

Using the API

arnold@peeringdb.com

Agenda

- Introduction
- JSON
- HTML Operations
- Record Types
  - Basic Records
  - Derived Records
- *The jq JSON Postprocessor*
- *Examples*

Introduction

- Why API (Application Programming Interface)?
  - The GUI is nice for human beings
  - Automation needs structured data
- Makes it easy to integrate PeeringDB in your environment

JSON

- Open standard file format
- Short for JavaScript Object Notation
- Filenames use the extension .json
- Language independent data format
- Basic data types
  - Number
  - String
  - Boolean
  - Array
  - Object
  - null

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

Basics

- In general https://peeringdb.com/api/OBJ
  - OBJ is case insensitive
  - So called endpoint: /api/OBJ
- Output always fits in one object
  - Meta is optional
  - Data always an array

```
{
meta:
  {
  status:
  message:
  }
data:
  [
  {},
  {}
  ]
}
```

Authentication

- Authentication via basic HTTP authorization
- Guest access does not need any authentication
- Examples
  - curl -sG https://username:password@peeringdb.com/api/poc
  - curl -u username:password https://peeringdb.com/api/poc
  - Put credentials in .netrc
    - machine peeringdb.com login username password password
- Recap: only access to contact information may be restricted
  - Endpoint /api/poc

Operations

- All HTML operations are supported
  - GET
    - Requests a representation of the specified resource
  - POST
    - Used to submit an entity to the specified resource
  - PUT
    - Replaces all current representations of the target resource with the request payload
  - DELETE
    - Deletes the specified resource

GET

- GET
  - Multiple objects
    - Endpoint /api/OBJ
  - Single object
    - Endpoint /api/OBJ/id

Optional URL parameters for GET

- limit
  - Integer value
  - Limits to n rows in the result set
- skip
  - Integer value
  - Skips n rows in the result set
- depth
  - Integer value
  - Nested sets will be loaded
  - See Nesting slide

Optional URL parameters for GET

- fields
  - String value
  - comma separated list of field names
  - only matching fields will be returned in the data
- since
  - Integer value
  - Retrieve all objects updated since specified time
  - Unix timestamp in seconds
- *fieldname*
  - Integer or string value
  - Queries for fields with matching value

Nested Data / Depth

- Of type OBJ_set
- Example: *net_set* will hold network objects
- Depth (for endpoint /api/OBJ)
  - 0: don't expand anything (default)
  - 1: expand all first level sets to ids
  - 2: expand all first level sets to objects
- Depth (for endpoint /api/OBJ/id)
  - 0: don’t exand anything (default)
  - 1-4: expand all sets and related objects according to level of depth specified

Query modifiers

- numeric fields
  - __lt: less than
  - __lte: less than equal
  - __gt: greater than
  - __gte: greater than equal
  - __in: value inside set of values (comma separated)
- string fields
  - __contains: field value contains this value
  - __startswith: field value starts with this value
  - __in: value inside set of values (comma separated)

POST

- Used to create an object
- Endpoint /api/OBJ
- Required parameters
  - Depending on OBJ
  - For *org* you need the *name*
  - For *fac*, *ix*, *net* you need the *org_id*
  - for *fac* you need the *name*
  - For *ix* you need the *name*
  - For *net* you need the *asn*

```
{
  "name": "Org-22106"
}
```

File 22106.json

- Example
  - curl -sn -X POST -H "Content-Type: application/json" -d @22106.json \
    https://tutorial.peeringdb.com/api/org

PUT

- Used to edit object
- Endpoint /api/OBJ/id
- Updates data in OBJ/id

```
{
        "name": "Org-22106",
        "address1": "23 Mulholland Drive",
        "city": "Los Angeles",
        "country": "US"
}
```

File 22106.json

- Example
  - curl -sn -X PUT -H "Content-Type: application/json" -d @22106.json \ https://tutorial.peeringdb.com/api/org/22114

- Operation of PUT is idempotent

DELETE

- Used to delete objects
- Endpoint /api/OBJ/id
- Example
  - curl -sn -X DELETE -H "Content-Type: application/json“ \
    https://tutorial.peeringdb.com/api/org/22114

Object Types

- Basic Objects
  - org, fac, ix, net, poc
- Derived Objects
  - ixlan, ixpfx, netixlan, netfac

Basic Objects

- org
  - Root object for fac, ix, net
  - Holds information about organisation
- fac
  - Describes a facility / colocation record
  - More useful information are in derived records netfac
- ix
  - Describes an Internet Exchange
  - More useful information are in derived records ixlan, ixpfx and netixlan
- net
  - Describes a network / ASN
  - More useful information are in netfac and netixlan
- poc
  - Describes various role accounts (point of contact)
  - Currently only for net objects

Derived Objects

- ixlan
  - Describes the LAN of an IX
  - One IX may have multiple ixlan
  - May go away with PeeringDB 3.0
- ixpfx
  - Describes the IP range (IPv4 and IPv6) for an ixlan
  - One ixlan may have multiple ixpfx
- netixlan
  - Describes the presence of a network at an IX
- netfac
  - Describes the presence of a network at a facility