PEERINGDB 2.0

MATT GRISWOLD

www.unitedix.net

UNITED
INTERNET
EXCHANGE

VERSION 1 ISSUES

- **Old, generated code, unmaintainable**
- **Schema issues**
  - 1 network per user, etc.
- **MySQL the only "API"**
  - Insecure; doesn't scale

VERSION 1 ISSUES

- No data validation
- Lots of typos
- Exposed contact information to potential spammers

VERSION 2

- New, clean, shiny python
- Completely redesigned schema
- RESTful API
- All data is cleaned and validated
- Contact info has permissions
  - Guest login won't see contact details

VERSION 2

- **Everything is permissioned and editable**
  - Allows data centers and IXs to update their own info
- **Documented APIs!**
- **New features planned after initial cut-over**
- **Beta version is live now at beta.peeringdb.com**

API SPECS

- **Each data type has an associated tag**
  - net
  - org
  - ix
  - etc

docs.peeringdb.com/api_specs »

API SPECS

- **To list all networks:**

```
curl -X GET https://<username>:<password>@beta.peeringdb.com/api/net
```

- **To view a specific network:**

```
curl -X GET https://<username>:<password>@beta.peeringdb.com/api/net/20
```

docs.peeringdb.com/api_specs »

API SPECS

- **All operations are supported**
  - read
  - write
  - create
- See docs for further details

docs.peeringdb.com/api_specs »

PYTHON LIBRARY

- **Very early in life cycle**
  - Expect more tests and features in the near future
- **Python seems to be the go-to language for network people**
- **More languages and libraries will show up, PHP will probably be next**

github.com/peeringdb/peeringdb-py »

PYTHON LIBRARY

- **Advantages**
  - Local (not dependent on servers being up, etc.)
  - Custom indexes can be built
  - Custom fields can be added
  - Database engine can be chosen (MySQL, Postgres, SQLite
- **To install:**

```
pip install peeringdb
```

github.com/peeringdb/peeringdb-py »

PYTHON LIBRARY

- **To configure a local database:**

```
peeringdb configure
```

- **To keep in sync after configuration:**

```
peeringdb sync
```

github.com/peeringdb/peeringdb-py »

PYTHON LIBRARY

COMMAND LINE INTERFACE

- To output YAML:

```
peeringdb get net20
```

- To output JSON:

```
peeringdb get -O json net20
```

github.com/peeringdb/peeringdb-py »

PYTHON LIBRARY

- **To build custom applications with the PeeringDB client library:**

```
from peeringdb.client import PeeringDB
  pdb = PeeringDB()
```

github.com/peeringdb/peeringdb-py »

DJANGO-PEERINGDB

- Django module
- Easy to integrate in a common web framework
- Multiple database options
- Used by peeringdb-py to sync data

DJANGO-PEERINGDB

- **Use cases**
  - Very easy to generate peering router config

```
protocol bgp chix_{{tag}} {
        description "Peer: as{{peer.asn}} ({{peer.descr}})";
        neighbor {{peer.ip}} as {{peer.asn}};
        route limit {{peer.max_prefix}};
```

UNITED IX SIGNUP

- **Customer signs up, gives ASN, backend system queries and auto populates IXP Manager data**
  - NOC info
  - Max prefix

OTHER COMPANIES USING V2 INTERFACE

- **Couchbase sync**
- **Netflix**
  - Redis sync
  - github.com/netflix/peeringdb-py »

EXAMPLE SCRIPT

https://github.com/grizz/pdb-examples

```
# search by ix and name
ix = pdb.all('ix', name='chix', country='us')[0]
pprint(ix)
```

EXAMPLE SCRIPT

https://github.com/grizz/pdb-examples

```
# get lan on ix
ixlan = pdb.all('ixlan', ix_id=ix['id'])[0]
pprint(ixlan)
```

EXAMPLE SCRIPT

https://github.com/grizz/pdb-examples

```
# lookup network by ASN
net = pdb.all('net', asn=63311)[0]
pprint(net)
```

EXAMPLE SCRIPT

https://github.com/grizz/pdb-examples

```
# lookup peer info by network and ixlan
peerings = pdb.all('netixlan', ixlan_id=ixlan['id'],
asn=63311)
pprint(peerings)
```

PEERINGDB 2.0

MATT GRISWOLD

www.unitedix.net

UNITED
INTERNET
EXCHANGE