Local Installations - Peering Setup

Peering Setup - Local Installations:

ProVision versions 8.2 and newer

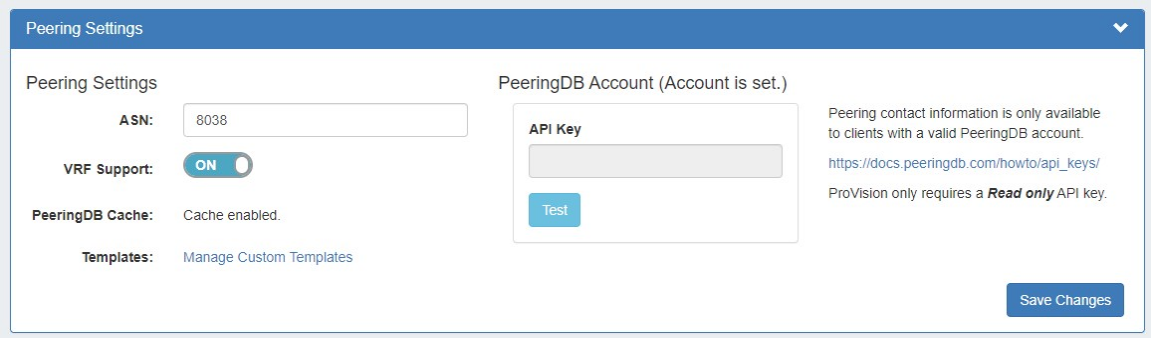
PeeringDB Options:

Local Installations may opt to sync peering data via the PeeringDB API (recommended), or by self-managing a local database utilizing a cron task to sync data - this option is only recommended for advanced users experiencing heavy loads.

Sync via PeeringDB API (Recommended):

> ProVision only requires a ***Read Only*** API Key. For information on PeeringDB API keys, please refer to: https://docs.peeringdb.com/howto/api_keys/

To set a PeeringDB Account, go to Admin Settings Peering, and enter the Read Only PeeringDB API key. Click "Test" to verify the connection.



When done, click "Save Changes". If successful, the phrase "Account is set" will display above the API key form.

Self-Manage a Local DB (Not Recommended):

Instead of making API requests to PeeringDB, you can now make them to a local PeeringDB copy and sync through either a separate cron job or by using the associated Scheduler Task.

> This option is only recommended for advanced users experiencing heavy loads. Most users should utilize the API Key option, above.

Define Constants

To link a provision instance to PeeringDB, define these constants in globals.php, where the values match the actual database that is hosting PeeringDB. :

```
define('PEERINGDB_HOST', 'localhost');

define('PEERINGDB_USERNAME', 'root');

define('PEERINGDB_PASSWORD', 'mypass');

define('PEERINGDB_NAME', ‘peeringdb');
```

This will change Settings section in the Peering area so it will now show the database status, instead of asking you for an account API KEY.

Installation:

For local customers, they can find install instructions for the PeeringDB Python Client here:

https://peeringdb.github.io/peeringdb-py/

You may create a config file as follows:

```
$ cat .peeringdb/config.yaml
orm:
  backend: django_peeringdb
  database:
    engine: mysql
    host: localhost
    name: peeringdb
    password: supers3cr3t
    port: 0
    user: peeringdb
  migrate: true
  secret_key: ''
sync:
  api_key: YOUR_API_KEY
  only: []
  password: ''
  strip_tz: 1
  timeout: 0
  url: https://www.peeringdb.com/api
  user: ''
```

Sync set up:

To run a sync of PeeringDB:

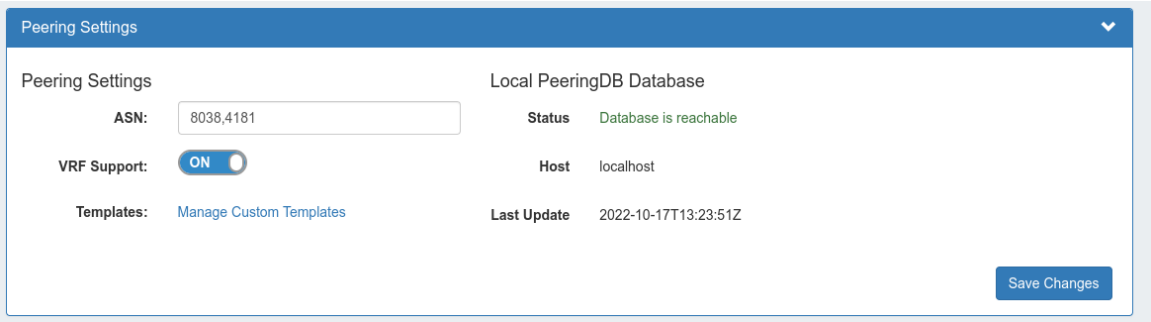
```
. pdbvenv/bin/activate
peeringdb sync
```

The database can also be synced using the scheduler. The task needs to know the location of the python virtual environment where peeringdb-py is installed and the directory that hosts the peeringdb-py config file.
For information on how to add a scheduler task, see Scheduler Tab.

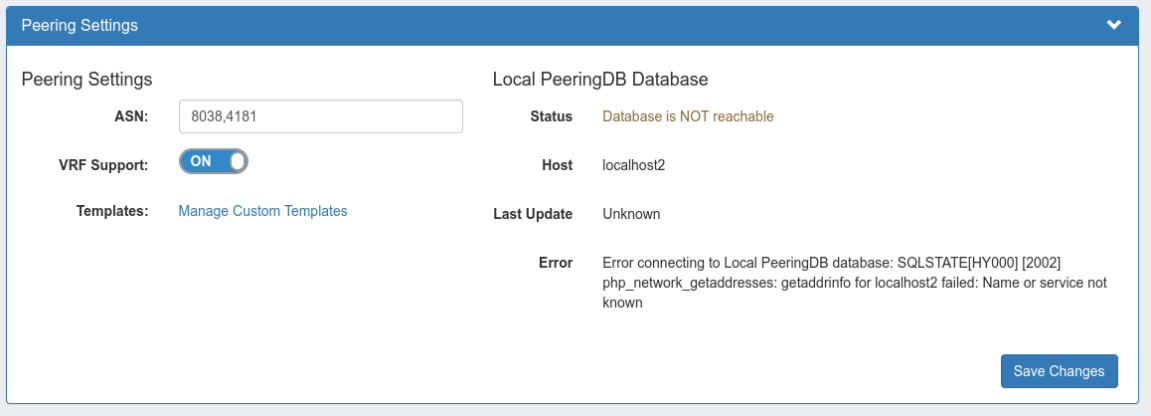
Customers who are not using a python virtual environment, or who are not running their local peeringdb database on the same server, should be able to set up their own cron task as outlined in the documentation below:
https://peeringdb.github.io/peeringdb-py/cli/#sync

View the Status:

Once set up, If the local database is reachable a green success message will display in Admin Settings Peering Settings.



If the database fails to connect, a warning and error message will display instead:



ProVision versions 7.x and newer

For ProVision versions 7.x and later, ProVision directly interfaces with PeeringDB's API to update exchange and peering data, caching the data for a default time of 12 hours.

This requires a PeeringDB account, and for the account credentials to be set in ProVision. The credentials may either be hard coded into globals.php, or set into the database via the Admin/Peering GUI. See Admin Settings and Peering for detailed information.

Additional Peering constants may be added into globals.php to change the PeeringDB URL between the main and beta site (some users may find the beta site to have faster response times), and to adjust the PeeringDB cache TTL.

For real-time updates, TTL may be set to 0. However, some users may experience severe lag with a TTL = 0; we recommend using a 10 to 15 minute or greater TTL if this occurs.

ProVision versions 5.3.0 to 6.1.2 (Legacy)

For legacy set up (not recommended):

ProVision uses a locally-hosted mirror of the PeeringDB database in order to perform non-edit Peering functions. There are a few steps to take in order to set up your locally hosted instance to coordinate with PeeringDB information.

> As of PeeringDB 2.0, SQL dump files are no longer provided. If you are using ProVision 5.3.0 or higher, you must follow this new install process. If you are using a lower version of ProVision, then please follow the instructions in the previous version of this page.

1) Create a new database to store the PeeringDB data. This **must** be on the same server as the database which is used by ProVision.

2) Download, install, and use the PeeringDB Python Client to populate the database.

The PeeringDB documentation is available here: http://peeringdb.github.io/peeringdb-py/

3) Once this has been done, edit the ProVision global configuration file located here:

```
[ProVision Root]/data/globals.php
```

It must be updated with the following variables to inform ProVision of the location of this new install. The username and password fields correspond to the username and password of the MySQL account which has access to the database (Not the username and password to your PeeringDB account).

> This can, but does not have to be, the same MySQL user which is used for the ProVision database. However, the ProVision MySQL user **must** have at least READ access to the PeeringDB database.

```
$peeringdb_host = 'localhost';          // Database host must be the same for provision and peering
$peeringdb_username = 'username';       // username for the MySQL user
$peeringdb_password = 'password';       // password for the MySQL user
$peeringdb_db_name = 'peeringdb';       // name of the database in MySQL
```

4) Periodically sync with the PeeringDB server to get the latest updates. This can be done manually, or there are instructions in the PeeringDB documentation on how to automatically schedule syncs using cron (http://peeringdb.github.io/peeringdb-py/cli/#sync).

Peering Constants

In globals.php, the following constants can be defined to tweak the Peering internals:

PEERINGDB_USERNAME

```
define('PEERINGDB_USERNAME', 'username');
```

Default value: none

The username for the account used to connect to the PeeringDB API

Instead of saving the username and password in the database, the values can be hard coded into globals.php

PEERINGDB_PASSWORD

```
define('PEERINGDB_PASSWORD', 'mypass');
```

Default value: none

The password for the account used to connect to the PeeringDB API

PEERINGDB_URL

```
define('PEERINGDB_URL', 'https://www.peeringdb.com/api/');
```

Default value: https://www.peeringdb.com/api/

The URL of the PeeringDB API. Alternate value: https://beta.peeringdb.com/api/

Note: PeeringDB URL Change - (Affects only v8.0.1 and earlier)

Previous versions of PeeringDB allowed authenticated requests to `https://peeringdb.com/api/.` As of March 2022, PEERINGDB_URL must be set to https://www.peeringdb.com/api/ for authenticated requests to succeed.

Customers updated to v8.0.2 and later do not need to take any action and are not affected.

Local customers using versions 8.0.1 or earlier, and who are using Peering (or wish to start using Peering), should perform the following update(s) as applicable to their situation:

A) Existing local customers who already have their peeringdb account details saved in Provision should update PEERINGDB_URL in globals.php to:

```
define('PEERINGDB_URL', 'https://www.peeringdb.com/api/');
```

B) Customers who wish to begin using Peering (thus, do not already have saved peeringdb account details) will not be able to add their account via the GUI prior to v8.0.2. An attempt to do so will fail the "valid account test", which uses a hardcoded string value.

Instead, affected users must do both of the following:

B1) Save peeringdb account details using the API:

```
https://[$your instance]/APIv2/spec.php?family=peering#/default/save_account
```

B2) Update PEERINGDB_URL in globals.php to:

```
define('PEERINGDB_URL', 'https://www.peeringdb.com/api/');
```

PEERINGDB_CACHE_TTL

```
define('PEERINGDB_CACHE_TTL', 43200);
```

Default value: 43200 (12 hours)

How often (in seconds) to purge the cached PeeringDB API calls. If a customer wants real time access, this can be set to 0.

If experiencing major lag issues with real time access, it is recommended to increase to increase the cache TTL from 0 to 5, 10, or 15 minutes.

Additional Information:

For additional information on working with Peering, see the following documentation sections:

- Peering
- Import Peering Sessions