# dnsupdate Documentation

***Release 0.4***

**Ben Wolsieffer**

**Nov 21, 2020**

# CONTENTS

CHAPTER

ONE

# CONFIGURING DNSUPDATE

## 1.1 Configuration File

**dnsupdate** is configured using a single YAML configuration file. This file can specified on the command line, or placed at either `~/.config/dnsupdate.conf` or `/etc/dnsupdate.conf`.

The available options are documented below.

### 1.1.1 address_provider

Controls how **dnsupdate** obtains IP addresses. If this option is not specified, a web service is used. When specified at the root of the file, this option applies to all DNS services.

Internally, address providers are Python classes which are initialized using this configuration option.

```
address_provider:
    type: ClassName;
    args:
        arg1: value1;
        arg2: value2;
```

`type` specifies the name of the class, while `args` is a list of constructor arguments.

Optionally, a shorthand notation can be used, which allows you to directly call the class's constructor:

```
address_provider: ClassName(value1, value2)
```

In this case, the option value is simply passed to `eval()`, so it is possible to evaluate arbitrary Python code.

Address providers can be specified separately for IPv4 and IPv6 in this manner:

```
address_provider:
    ipv4:
        type: ...
        args:
            ...
    ipv6:
        type: ...
        args:
            ...
```

If only one of the two protocols is configured, the other protocol is disabled (unless a specific service overrides this option).

Default: *Web()*

### 1.1.2 dns_services

A list of services to update. Each service is represented internally as a Python class. The arguments specified in the configuration are directly passed to the class's constructor.

It is also possible to override the global address provider for a specific entry, using the same format as the global `address_provider` option. If only one address provider protocol (IPv4 or IPv6) is specified, the other is inherited from the global configuration. It is possible to disable a protocol that was configured globally by assigning it the value `None`. See `examples/provider_override.conf`.

```
dns_services:
    - type: ServiceClassName1
      address_provider: <see above>
      args:
          arg1: value1
          arg2: value2
    - type: ServiceClassName2
      args:
          arg1: value1
          arg2: value2
    ...
```

The shorthand constructor notation can also be used to initialize a DNS service.

### 1.1.3 cache_file

Path to the file where **dnsupdate** will store information about the configured DNS services, such as their addresses and whether they are enabled. The specified file must be writable by **dnsupdate**.

Default: `~/.cache/dnsupdate.cache`

## 1.2 Address Providers

**class Web**(*ipv4_url='https://ipv4.icanhazip.com/'*, *ipv6_url='https://ipv6.icanhazip.com/'*)

Retrieves addresses from a web service (by default: icanhazip). This provider expects the response to contain only the address in plain text (no HTML).

**Parameters**

- **ipv4_url** – URL of the service that retrieves an IPv4 address
- **ipv6_url** – URL of the service that retrieves an IPv6 address

**class Local**(*interface*, *allow_private=False*)

Retrieves addresses from a local network interface. If you are behind NAT (which is often the case if you are using dynamic DNS), this provider will return not return any IPv4 address, unless you enable the `allow_private` option. Normally, you will want to use a different provider for IPv4 if you are behind NAT.

**Parameters**

- **interface** – name of the interface to use
- **allow_private** – consider a private address to be valid

**class ComcastRouter**(*ip*, *username='admin'*, *password='password'*)

Scrapes the external IPv4 address from a Comcast/XFINITY router. This address provider does not support IPv6 because it doesn't usually make sense to submit the router's IPv6 address to a dynamic DNS service.

This has been tested with an Arris TG1682G, but may work with other routers using Comcast's firmware.

**Parameters**

- **ip** – internal IP address of the router
- **username** – username for the web interface (usually 'admin')
- **password** – password for the web interface (router default is 'password')

## 1.3 DNS Services

**class StandardService**(*service_ipv4*, *service_ipv6*, *username*, *password*, *hostname*, ***extra_params*)

Updates a DNS service that uses the defacto standard protocol that has been defined by Dyn. All the standard return codes are handled. Client configuration errors will cause the service to be disabled.

**Parameters**

- **service_ipv4** – domain name of the IPv4 update service
- **service_ipv6** – domain name of the IPv6 update service
- **username** – service username (some services use the your (sub)domain name as the username)
- **password** – service password (sometimes this is a unique password for a specific (sub)domain rather than your actual password)
- **hostname** – fully qualified domain name to update
- **extra_params** – other keyword arguments that will be appended to the request URL

**class FreeDNS**(*ipv4_key*, *ipv6_key=None*)

Updates a domain on FreeDNS using the version 2 interface. This API uses a single key for each entry (separate ones for IPv4 and IPv6) instead of separately passing the domain, username and password. The keys are the last part of the given update URL, not including the trailing slash.

For example, the update key for the URL `http://sync.afraid.org/u/VWZIcQnBScVv8yv8DhJxDbnt/` is `VWZIcQnBScVv8yv8DhJxDbnt`

**Parameters**

- **ipv4_key** – update key for IPv4
- **ipv6_key** – update key for IPv6

**class NSUpdate**(*hostname*, *secret_key*)

Updates a domain on nsupdate.info. nsupdate.info uses the Dyn protocol.

**Parameters**

- **hostname** – fqdn to update
- **secret_key** – update key

**class OVHDynDNS**(*username*, *password*, *hostname*, *system='dyndns'*)

Updates a domain using OVH's DynDNS service. This service uses the standard Dyn protocol, with an extra `system` parameter. This service only supports IPv4.

**Parameters**

- **username** – account username
- **password** – account password

- **hostname** – the hostname to update
- **system** – the type of update (default: `dyndns`)

**class GoogleDomains**(*username*, *password*, *hostname*)

Updates a domain registered through Google Domains. This service uses the standard Dyn protocol. This service only supports IPv4.

**Parameters**

- **username** – account username
- **password** – account password
- **hostname** – the hostname to update

**class StaticURL**(*ipv4_url*, *ipv6_url=None*)

Updates addresses by sending an HTTP GET request to statically configured URLs. When using this type of service, the remote server will automatically detect your IP and therefore the address provided by the configured address provider will not be used. In most cases, the result will be the same as if the *Web* provider had been used.

**Parameters**

- **ipv4_url** – URL used to update the IPv4 address
- **ipv6_url** – URL used to update the IPv6 address

CHAPTER

TWO

# USING DNSUPDATE

Dynamic DNS update client

```
usage: dnsupdate [-h] [-f] [-V] [config]
```

## 2.1 Positional Arguments

**config** the config file to use

## 2.2 Named Arguments

**-f, --force-update** **force an update to occur even if the address has not changed** or a service has been disabled

**-V, --version** show program's version number and exit

**dnsupdate** is designed to be run as a cron job or with any other scheduler. It checks for address changes and exits after sending any necessary updates. Example systemd service and timer files are included in the root of the repository. The service file is automatically installed as part of the Arch Linux AUR package.

On startup, **dnsupdate** checks if the addresses for any of the configured services have changed, and if so it will attempt to update them. If an update fails and the service reports that the problem was due to client misconfiguration (such as an incorrect password or hostname), the service will be disabled until the config file is edited.

The `-f` flag can be used to force the update of all services, even if no addresses have changed or a service is disabled. This flag should not be used as part of the automatic update process because too many update attempts that result in no change will cause some services to ban you.

CHAPTER

# THREE

# EXTENDING DNSUPDATE

**dnsupdate** is designed to make it easy to add new address providers and DNS services.

If you add a new address provider, DNS service or any other feature that might be useful to others, feel free to submit a pull request on GitHub.

## 3.1 Adding an address provider

To add support for a new address provider, add a new subclass of *AddressProvider* to the the module, and it will become available to use in a configuration file.

**class AddressProvider**

Provides a standard interface for retrieving IP addresses. Any information needed to obtain the addresses (such as authentication information) should be specified in the constructor. Constructor arguments will become options that can be specified in the config file.

Implementations must use the requests library for all HTTP requests. This should be done by calling methods of the `session` variable in this module, rather than calling the global `requests` functions. This makes sure all requests have the correct user agent.

**ipv4**()

Return an IPv4 address to assign to a dynamic DNS domain. Only implement this method if your address provider supports IPv4.

**Return type** `ipaddress.IPv4Address`

**ipv6**()

Return an IPv6 address to assign to a dynamic DNS domain. Only implement this method if your address provider supports IPv6.

**Return type** `ipaddress.IPv6Address`

## 3.2 Adding a DNS service

Likewise, a new DNS service can be created by subclassing *DNSService*.

**class DNSService**

Provides a standard interface for updating dynamic DNS services. Any information needed to perform an update (such as the domain name and password) should be specified in the constructor. Constructor arguments will become options that can be specified in the config file.

If possible, implementations should send the *address* parameter of the update methods to the service, rather than letting the service automatically detect the client's address. This makes it possible (with a custom address provider) for a client to point a domain at another device.

Implementations must use the requests library for all HTTP requests. This should be done by calling methods of the `session` variable in this module, rather than calling the global `requests` functions. This makes sure all requests have the correct user agent.

To indicate errors during the update process, implementations of the update methods can raise one of three special exceptions: *UpdateException*, *UpdateServiceException* or *UpdateClientException*. See the documentation for these classes for information on when they should be raised.

**__str__**()
If possible, implement this function to provide more information about the service, such as the hostname. The recommended format is `ClassName [hostname, etc]`. Use `self.__class__.__name__` for the class name rather than hard-coding it.

**update_ipv4**(*address*)
Update the IPv4 address of a dynamic DNS domain.

**Parameters** **address** (`ipaddress.IPv4Address`) – the new IPv4 address

**update_ipv6**(*address*)
Update the IPv6 address of a dynamic DNS domain.

**Parameters** **address** (`ipaddress.IPv6Address`) – the new IPv6 address

### 3.2.1 Update exceptions

**class UpdateException**
Signals that an error has occurred while attempting to perform an address update, but the client does not know whether it was caused by a misconfiguration or a problem with the service. If the reason for the error is known, one of this exception's subclasses should be used instead.

**class UpdateClientException**
Signals that an error has occurred as the result of a misconfiguration of the client. This exception should only be raised when there is very little chance that an error occurred due to a temporary problem with the DNS update service. The reason for this is that when this exception is thrown, **the service that was being updated will be disabled until the user edits the configuration file**.

**class UpdateServiceException**
Signals that an error has occurred on the DNS service's server while attempting an update. In this case, an error will be printed, but the service will not be disabled.

- genindex

# PYTHON MODULE INDEX

## d

# INDEX