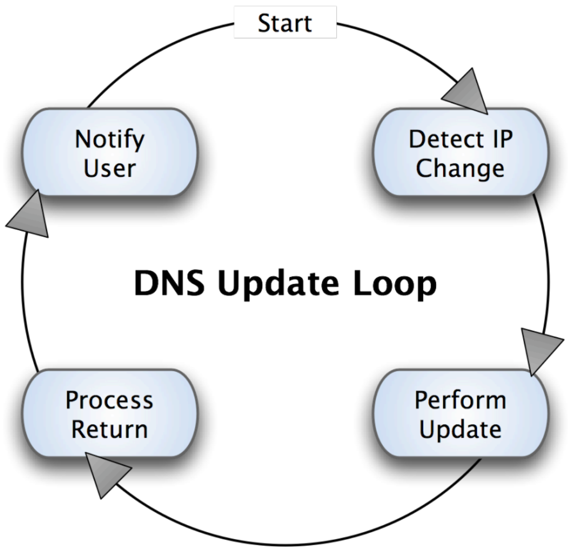
DNS Update API

November 15, 2006 – Version 2.0.3



Dynamic Network Services, Inc.
1230 Elm Street, Fifth Floor
Manchester, NH 03101

phone: +1-603-668-4998
fax: +1-603-668-6474
www.dyndns.com

Table of Contents

DNS Update API

The DNS Update API is also known as the Members NIC Update API and is used to update the IP addresses of dynamic DNS hostnames. DynDNS designed and created the specification prior to 2001, which has become the standard update mechanism for other DNS providers. The API works over HTTPS and HTTP.

As this update mechanism has been integrated into numerous products, any changes to the API will be completely backwards compatible.

Developers who have integrated the DNS Update API into their devices are encouraged to enhance their products through the DynDNS Certified Program.

DynDNS also offers Private Label DynDNS for developers and companies looking for an integrated DNS platform.

DNS Update Loop

The client's purpose is to keep a hostname up to date with a user's current IP address. This is done with in a loop:

- Detect IP Change – Check for changes to the current IP address
- Perform Update – If the IP address has changed or a user updates any setting
- Process Return – Parse return code
- Notify User – Perform logging, notify the user (if necessary), shutdown on fatal errors

Flow Diagram

A flow diagram, included in this document, provides an overview of the behavior we require from clients. This document is intended to supplement the other specification documents.

Detect IP Changes

To determine whether a client should update, it must have a reliable method to determine what its current IP address is so it can compare the current address to last updated address. There are two methods.

Directly Connected

A client may determine, automatically or through the input of the user, that it is directly connected to the Internet. The device would have a publicly addressable IP address. In this case, the optimal method is to use API methods from the parent operating system's network stack.

Web IP Detection (CheckIP)

A client may determine, automatically or through the input of the user, that it is not directly connected to the Internet. The client is on a machine with a private IP, usually on RFC 1918 space (10/8, 172.16/16, 192.168/16). In this case, the optimal method is to use web IP detection.

Clients in this setup are typically behind a NAT and will need to setup port forwarding.

DynDNS operates a CheckIP tool, accessible at http://checkip.dyndns.com/ that may be used with clients that work with DynDNS.

Complications

Initialization

When a client starts for the first time, it is expected that it will perform an update when it is first enabled. This may result in a `"nochg"` return code which increments an abuse counter. Great care should be taken to prevent initialization unless it is necessary.

It is expected that the IP address will be stored in a non-volatile manner. This is especially true for hardware-based devices.

Multiple Interfaces

Some clients can be installed on devices with multiple network interfaces, such as a gateway or router. If this is the case, the developer needs to base a basic assumption or the user needs to be given the option as to which interface to use for IP detection.

Check IP Tool

DynDNS offers a free web IP detection tool for use with DynDNS services. Check IP will return the remote socket's IP address. If a client sends a `Client-IP` or a `X-Forwarded-For` HTTP header, Check IP will return that value instead.

Hostname: http://checkip.dyndns.com

Call and Answer

Check IP responds to good HTTP requests for `http://checkip.dyndns.com/`. A valid request will result in the following sample response:

```
HTTP/1.1 200 OK
Content-Type: text/html
Server: DynDNS-CheckIP/0.1
Connection: close
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 105

<html><head><title>Current IP Check</title></head><body>Current IP Address:
123.456.78.90</body></html>
```

Policies

- Use the OS if possible for IP detection

- Checks must be spaced 10 minutes apart to help reduce server load
- In the case of an error while accessing Check IP, the client should not send an update

Perform Update

When a change in IP address is found or a user alters any of their settings, the client should perform an update. All updates are sent using a well-formed HTTP request. DynDNS will pass back a return code that the client needs to parse. The update API is a REST-based system.

The HTTP Request

Updates can be performed over HTTP or SSL-encrypted HTTPS (preferred).

Update hostname:	members.dyndns.org
HTTP update ports:	80, 8245
HTTPS update port:	443

All requests should be sent to `members.dyndns.org`. The client, or the underlying OS, will need to perform a DNS query and appropriately cache the result. Hard coding the IP address is not acceptable as the IP address may change.

The update interface listens on ports 80 and 8245 for HTTP, and 443 for HTTPS. Port 8245 may be used to bypass transparent HTTP proxies. It is not necessary to open any incoming ports (or allow incoming ICMP) for updating.

All clients must send a well-formed user agent that includes company name, model number, and software build revision. An example would be: `Company Name - Device DNSUPDATE - 3.2`

Examples

The examples are provided only as samples and be familiar with RFC 2616 for information about the HTTP Protocol.

- These are the bare minimum headers.
- "username:pass" must be encoded in base64.

Example 1: Authentication in URL (all one line)

```
http://username:password@members.dyndns.org/nic/update?
        system=dyndns&
        hostname=yourhost.ourdomain.ext,yourhost2.dyndns.org&
        myip=ipaddress&
        wildcard=NOCHG&
        mx=NOCHG&
        backmx=NOCHG&
```

Example 2: HTTP GET Request (all one line)

```
GET /nic/update?
    system=statdns&
    hostname=yourhost.ourdomain.ext,yourhost2.dyndns.org&
    myip=ipaddress&
    wildcard=OFF&
    mx=mail.exchanger.ext&
    backmx=NO&
    offline=NO
    HTTP/1.0
Host: members.dyndns.org
Authorization: Basic username:pass

User-Agent: Company - Device - Version Number
```

Update Parameters

Field	Data	Description	Default	dyndns	static	custom
system	dyndns|statdns|custom	The system you wish to use for this update. DynDNS will update a Dynamic DNS hostname, custom will update a Custom DNS hostname, while statdns will update a Static DNS hostname. If the system parameter given is not dyndns, custom or statdns, the update will fail. If the system parameter is not specified at all, it will default to dyndns.	dyndns	Y	Y	Y
hostname	host1.dyndns.org,host2.dyndns.org...	These are the hostnames you wish to update. Each hostname specified will be updated with the same return code for each host will be given one per line, in the same order as the hosts given. (e.g. $host1_return_code,$host2_return_code)	None, required field	Y	Y	Y
myip	ip_address	IP address to set for the update. If this parameter is not specified, the best IP address the server can determine will be used (some proxy configurations pass the IP in a header, and that is detected by the server). If the IP address passed to the system is	Remote socket IP address	Y	Y	Y

Field	Data	Description	Default	dyndns	static	custom
		not properly formed, it will be ignored and the system's best guess will be used.				
wildcard	ON|OFF|NOCHG	Enable/disable wildcards for this host (ON to enable). The wildcard aliases `*.yourhost.ourdomain.tld` to the same address as `yourhost.ourdomain.tld`.	OFF	Y	Y	N
mx	mailexchanger|NOCHG	Specifies a Mail eXchanger for use with the hostname being modified. The specified MX must resolve to an IP address, or it will be ignored. Specifying an MX of "NOCHG" will cause the existing MX setting to be preserved in whatever state it was previously updated via a client or the DynDNS website.	[clears any MX]	Y	Y	N
backmx	YES|NO|NOCHG	Request that the MX in the previous parameter be set up as a backup MX by listing the host itself as an MX with a lower preference value. A value of "NOCHG" will result in the previous value of the setting being preserved.	NO	Y	Y	N
offline	YES|NO	Sets the hostname to offline mode. This feature is only available to credited users. The `!donator` return will be used if the account is not credited.	NO	Y	N	Y

Return Codes

When updating a hostname, the response to the update syntax will be one of the return codes. If there is an error, clients should communicate to the user either a brief description of the problem that the return code indicates. To make troubleshooting easier, generic error messages such as "Unable to update" should not be used.

If updating multiple hostnames, hostname-specific return codes are given one per line, in the same order as the hostnames were specified. Return codes indicating a failure with the account or the system are given only once.

Update Types

Successful Updates

There is only one return code which indicates a successful update. A `"good"` indicates that the update was successful and that the IP address was changed in our system.

No Change Updates

A `"nochg"` indicates a successful update but the IP address or other settings have not changed. The only acceptable situation where is this allowed is during the clients if the host has already been set to the same IP address. Users may also be given the option to "force" an update. This can be used to verify the authentication credentials and the current IP address.

A this is fairly infrequent, repeated instances of `"nochg"` updates will result in the host being blocked. Users should be cautioned not to repeatedly force updates.

While it is not expected that the clients will prevent users from doing this, the client itself should strenuously avoid performing updates which would result in this return result.

Fatal Updates

Any failed update attempt is fatal which means that all further updates will also fail until the user has taken some sort of corrective action. For this reason, any failed update attempt should cause the client to be disabled until the situation is correct and the client re-enabled by the user.

Additionally, because the update may fail for a number of different reasons, the client needs to provide some method of communicating with the user that the update has failed and why. Some suggestions include:

- Logging a message in the general log window for the router (assuming it has one)
- Logging a message to a log window specific to the DynDNS client
- Generating an error message to an e-mail address configured by the user
- Communicating to an external process running on the users desktop

Many of these errors involve configuration mistakes within the client or inconsistencies between the client configuration and the user's account status; in all of those cases, the client must **stop** updating until the user has corrected the problem. Retrying the update automatically in those cases is abuse and may result in a client block.

Update Syntax Errors

The codes below should never be returned except when developing a client. They indicate a serious problem with the syntax of the update sent by the client.

`badsys`	The `system` parameter given is not valid. Valid system parameters are `dyndns`, `statdns` and `custom`.
`badagent`	The user agent that was sent has been blocked for not following these specifications, or no

	user agent was specified.

Account-Related Errors

The codes below indicate that the client is not configured correctly for the user's account. These return codes are given just once. The client must stop updating until the user confirms that the problem has been resolved.

`badauth`	The username or password specified are incorrect.
`!donator`	An option available only to credited users (such as offline URL) was specified, but the user is not a credited user. If multiple hosts were specified, only a single `!donator` will be returned.

Update Complete

The codes below indicate that the update of a hostname was completed successfully.

`good`	The update was successful, and the hostname is now updated.
`nochg`	The update changed no settings, and is considered abusive. Additional `nochg` updates will cause the hostname to become blocked.

Note that, for confirmation purposes, `good` and `nochg` messages will be followed by the IP address that the hostname was updated to. This value will be separated from the return code by a space.

Hostname-Related Errors

The codes below indicate a problem with a specific hostname. The client must stop updating that hostname until the user confirms that the problem has been resolved.

`notfqdn`	The hostname specified is not a fully-qualified domain name (not in the form hostname.dyndns.org or domain.com).
`nohost`	The hostname specified does not exist (or is not in the service specified in the system parameter)
`!yours`	The hostname specified exists, but not under the username specified.
`numhost`	Too many or too few hosts found
`abuse`	The hostname specified is blocked for update abuse.

If no hostnames were specified, `notfqdn` will be returned once.

Server Error Conditions

The codes below indicate server errors that will have to be investigated. The client must stop updating and ask the user to contact support. The client must not resume updating until the user confirms that the problem has been resolved, or a minimum of 1 hour has passed.

`dnserr`	DNS error encountered
`911`	There is a serious problem on our side, such as a database or DNS server failure.

The return `dnserr` will be followed by a numeric packet ID which should be reported to DynDNS Support along with the error.

Policies

To ensure fair use of our systems for the general public, client authors must ensure that clients follow these policies. Clients that do not comply with these policies may be blocked from accessing our systems.

Required Client Behavior

- Send a unique user agent which includes company name, model number, and software build revision.
- Check that all input is in valid form before updating.
- Check that any IP obtained through web-based IP detection is a valid dotted quad numeric IP (eg: 1.2.3.4) before sending it in an update.
- Only update when the IP address is different from the IP of the last update.

Unacceptable Client Behavior

- Send requests to or access anything other than `/nic/update` at the host `members.dyndns.org`.
- Reverse engineer web requests to our website to create or delete hostnames.
- Hardcode the IP address of any of DynDNS servers.
- Attempt to update after receiving the `!yours`, `notfqdn`, `abuse`, `nohost`, `nochg`, `badagent`, `badauth` or `badsys` return codes without user intervention.
- Perform DNS queries to determine whether the client IP needs to be updated.
- Access our web-based IP detection script (http://checkip.dyndns.com/) more than once every 10 minutes

Guidelines and Notes

These guidelines are our advice and recommendations on how to handle various issues involved in client development to ensure that the client does not violate our policies.

POST vs GET:

While this is not a requirement, GET should be used because debugging problems is easier.

System Startup and DHCP Leases

Clients should store the IP in some form of permanent storage, so that restarting the system (for software clients) or power cycling the device (for hardware clients) does not cause the client to update unless the IP is changed. Since IPs rarely change on DHCP lease renewals, clients must not update every time a DHCP lease is renewed. Instead they must check the new IP and determine if it differs from the old IP before updating.

DNS Queries

Clients should not perform DNS queries to determine whether it is necessary to update. The danger is that the ISP's DNS server will be caching the old IP for a few minutes, leading the client to conclude the update failed and causing a loop.

HTTP Headers

The HTTP headers returned may be status 200 (OK), 401 (Authorization Required) or 500 (Internal Server Error). The response body should be parsed for return codes no matter what this status is; a 911 return code will most likely have a HTTP status 500. The HTTP status will not indicate any particular message. Rely on the return codes instead.

Offline Redirection

When setting a host to offline mode, a `good` return code will be sent. The IP returned is that of the offline system, and should be ignored.

User Input

Users need to enter a username and password, each up to 16 characters long. They will also choose a system (dyndns, statdns, or custom) and enter a hostname. It is important that the hostname field be long enough (100–200 characters).

Client Blocks

Clients authors and developers who do not follow the DNS Update API may become blocked. Overly abusive clients harm DynDNS infrastructure and cause a suboptimal experience for other users. If a client poses a serious threat to DynDNS and a developer ignores a report of the problem, DynDNS will take any action necessary to block or otherwise disable those clients. If a client is targeted for a block, DynDNS will work with the developer as follows:

- Contact the developer/company to inform them of the problem
- Work to develop a fix as a firmware patch or future release
- Promote new software revision to encourage adoption

DynDNS Support

If you have questions about the DNS Update API, please contact Dynamic Network Services, Inc.

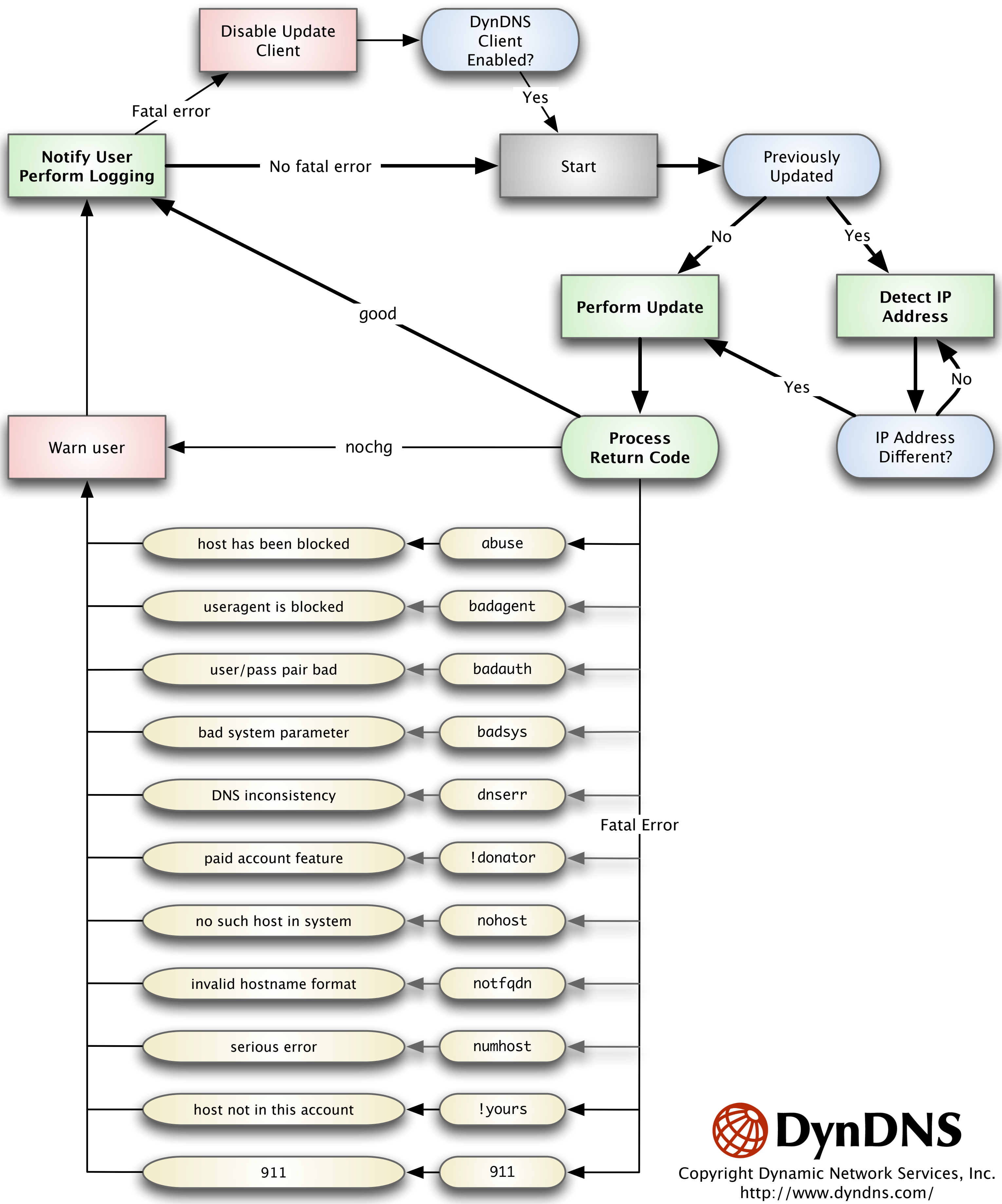
Dynamic Network Services, Inc.
1230 Elm Street, Fifth Floor
Manchester, NH 03101

+1-603-668-4998

support@dyndns.com

DynDNS Update API Flow Diagram

Last Updated: September 3, 2006



DynDNS