dynDNS.it - API

This document specifies the **dynDNS.it** API v3.0

The API is compatible with the original DNS Update API specification designed by Dyn in the late 1990s, which has become the standard update mechanism for dynamic DNS providers.

Important note: *Although the protocol is compatible with the dyndns.org protocol of dyn.com you must pay attention to the **specific endpoints** and **URLs** for the dynDNS.it service*

Dynamic 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 a loop

Step	Actions
1. Detect IP Change	Check for changes to the current IP address
2. Perform Update	If the IP address has changed or a user updates any setting or 24 hours have passed since the last update
3. Process Return Code	Parse return code after performing an update
Notify User	Perform logging, notify the user (if necessary), shutdown on fatal errors, delay on Server errors

In the following sections we will see a detail of all these steps.

1. Detect IP Change

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 the last updated address. There are two methods.

1. 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.

2. 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, 172.16/12, 192.168/16). In this case, the optimal method is to use web IP detection.

Impulso operates a **CheckIP service** which may be used with clients that work with DYNDNS.IT.

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 use a basic assumption or the user needs to be given the option as to which interface to use for IP detection.

CheckIP Service (Web IP Detection)

Impulso offers a free web IP detection tool for use with DYNDNS.IT service. CheckIP will return the remote socket's IP address.

If a client sends a Client-IP or a X-Forwarded-For HTTP header, CheckIP will return that value instead.

Endpoint: checkip.dyndns.it
HTTP port: 80
HTTPS port: 443

Call and Answer

CheckIP responds to valid HTTP or HTTPS requests for **http://checkip.dyndns.it/** or **https://checkip.dyndns.it/**. A valid request will result in the following sample response:

```
HTTP/1.1 200 OK
Date: Fri, 13 Sep 2024 10:01:49 GMT
Content-Length: 105
Content-Type: text/html; charset=UTF-8
Connection: close

<html><head><title>Current IP Check</title></head><body>Current IP
Address: 188.34.164.154</body></html>
```

Policies

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

2. Perform Update

All updates are sent using a well-formed HTTP or HTTPS request. The Web Service will pass back a return code that the client needs to parse.

When Sending an update request

The Client should perform an update:

- When the service is started (ie after turning on the device)
- When a change in IP address is detected
- When the user modifies any DDNS setting
- After 24 hours from the last update

The HTTP Request

Endpoint: update.dyndns.it
HTTP port: 80
HTTPS port: 443

All requests should be sent to update.dyndns.it. Hard coding the IP address is not acceptable as the IP address may change.

The update web service listens on ports 80 for HTTP, and 443 for HTTPS.

All clients must send a well-formed user agent that includes company name, model number, and firmware Version number and email contact.

Examples

These examples are provided only as samples. See RFC 2616 for information about the HTTP Protocol.

Authentication in URL

For web-browsers or utility programs (fetch, curl, lwp-request) that can parse the authentication section in URL.

```
https://{user}:{password}@update.dyndns.it/nic/update?hostname={hostname}&myip={IP Address}
```

or via http

```
http://{user}:{password}@update.dyndns.it/nic/update?hostname={hostname}&myip={IP Address}
```

Dyn.com v3 Authentication URL

```
http://{username}:{password}@update.dyndns.it/v3/update?hostname={
yourhostname}&myip={ipaddress}
```

Raw HTTP GET Request

Actual HTTP request should look like the following fragment. Note that there is the bare minimum set of headers. Requests should be followed by an empty line.

Fragment base-64-authorization should be represented by Base 64 encoded username:password string.

```
GET /nic/update?hostname=yourhostname&myip=ipaddress HTTP/1.0
Host: update.dyndns.it
Authorization: Basic base-64-authorization
User-Agent: Company - Device - Version maintainer@example.com
```

Basic base-64-authorization is a Base 64 encoded username:password string

Please note that POST requests are not permitted and will not be processed, don't use them.

Update Parameters

Field	Description	Example
`hostname`	Hostname that you wish to update. This field is required	hostname=example.homepc.it
`myip`	IPv4 address to set for the update. If this parameter is not specified, the best IP address the update service can determine will be used. If the IP address passed to the system is not properly formed, or is a private IP, it will be ignored and the system's best guess will be used.	myip=1.2.3.4

The web service answers with a Return Code, detailed in the next section.

3. Process Return Code

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 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.

Types of return codes

There are three different of return code types and the client should behave differently when receiving each of them:

Return Code Type	**Client behaviour**
Successful Update	The client should continue running, sending an update on the next IP change or after 24 hours from the last update.
Failed Update	The client should stop and send an update when the customer modifies the client configuration.
Server Error	The client should continue running postponing the update for 10 minutes at least.

Successful Update

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

Code	description	example
good	The update was successful, and the hostname is now updated.	`good 1.2.3.4`
nochg	The update changed no settings, and is considered abusive. To prevent this, the client should perform updates only when IP changes.	`nochg 1.2.3.4`

*Note that, as in the examples, **good** and **nochg** codes 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.*

Failed Update

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 corrected and the client is manually 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. For example showing a message in the DDNS configuration page or logging the error in the modem/router log.

Code	description	example
badauth	The username and password pair do not match a real user.	`badauth`
notfqdn	The hostname specified is not a fully-qualified domain name	`notfqdn`
abuse	The hostname specified is blocked for abuse.	`abuse`
nohost	The hostname specified does not exist in this user account	`nohost`

Server Error Conditions

The codes below indicate that the web service can be under maintenance. The client can continue running but delay the next update for 10 minutes.

Code	description	example
911	There is a problem or scheduled maintenance on our side.	`911`