

[UPnP workings](#)[UPnP IGD hacking](#)[IGD stacks](#)[Vulnerable IGD devices](#)[Possibly vulnerable IGD devices](#)[IGD Annoyances](#)[Getting access to DNS with UPnP](#)[UPnP A/V hacking](#)[UPnP RemoteUI hacking](#)[Unresearched UPnP hacks](#)[Downloads](#)[News](#)[Media](#)[Frequently Asked Questions](#)[Contact](#)[Links](#)

UPnP history

Universal Plug and Play (UPnP) saw the light in the late 1990s. Networks were just becoming popular. Several vendors were coming up with solutions to make networks and networked applications easier to manage. One early attempt was Sun's JINI. As a reaction to JINI (or so I was told) Microsoft came with UPnP. The first Microsoft products to ship with UPnP were Windows Millenium Edition and Windows XP. Since then there have been a lot of programs and devices that depend on UPnP (Live Messenger, Playstation, X-Box) and millions of networked devices that have implemented UPnP, such as routers and, increasingly, media players and media servers.

Early versions of the Microsoft UPnP software suffered from a few buffer overflows. Until 2006 these were the most widely known UPnP bugs. In 2006 at the SANE 2006 conference in Delft, the Netherlands, I presented a paper about bugs in other UPnP devices, which are hard to fix and detect for normal users. In January 2008 the GNUcitizen hacker group used a flaw in the Adobe Flash plugin for Internet Explorer to reconfigure routers with UPnP (but only some stacks) and turned a (mostly) local attack into a remote attack.

With more UPnP enabled devices on the market, and more people taking desktop security serious (well, to some extent) some of the focus is shifting towards other devices on the network, such as access points, routers and firewalls, although at the moment it seems that right now desktops are still the prime targets. I have the feeling this will change in the future.

What is UPnP?

The main goal of UPnP is to make adding network devices and networked programs to a network as easy as it is to plug in a piece of hardware into a PC (or even easier, as that is often error prone). The devices and programs find out about the network setup and other networked devices and programs through discovery and advertisements of services and configure themselves accordingly. In short: UPnP is a framework to build networked applications.

The use of the name UPnP has caused a lot of confusion. Product specifications often mention something like 'UPnP support', but are totally unclear about what kind of support. Technically, just implementing device discovery would make a product UPnP compatible.

Depending on the context 'UPnP' can mean completey different things. For a router this often means that the Internet Gateway Device Profile is implemented. For a media device it means that MediaServer, MediaRenderer or RemoteUI is implemented.

UPnP stack layout

The UPnP stack consists of 6 layers, one of which is optional:

1. Discovery
2. Description
3. Control

specifically the URLs that the control and eventing phase should send commands to, but possibly also other meta information about a device, such as an icon that should be displayed by Windows Explorer, the manufacturer of the device, and so on.

There is no default value for this header. In fact, in some devices, especially based on a Broadcom chipset, it is set dynamically at boot time. The only way to be completely sure is to always do device discovery.

Control

The third step in the protocol is "control": a device or program can ask another device or program to perform an action on the client's behalf, using SOAP. SOAP is a protocol that runs over HTTP and uses XML to describe remote procedure calls to a server and return results from those calls. SOAP is mainly used for web based services. For every major programming language libraries are available that can be used to implement SOAP requests and process SOAP responses.

Requesting a service is done by sending a SOAP request to the so called "control URL" of the control point, with the right parameters. The control URL for a specific profile can be found inside the <service> tag in the XML file found at the URL in the **LOCATION** header from the Description stage. The <service> tag from the Thomson Speedtouch 510 for the **WANPPPCConnection** profile looks like this:

```
<service>
  <serviceType>urn:schemas-upnp-org:service:WANPPPCConnection:1</serviceType>
  <serviceld>urn:upnp-org:serviceld:wanpppc:pppoa</serviceld>
  <controlURL>/upnp/control/wanpppcpppoa</controlURL>
  <eventSubURL>/upnp/event/wanpppcpppoa</eventSubURL>
  <SCPDURL>/WANPPPCConnection.xml</SCPDURL>
</service>
```

For sending SOAP requests only the URL inside the controlURL tag is necessary. It depends on the profile which actions can be performed. The URL found at the URL in the SCPDURL tag is the so called "URL for service description". It describes which SOAP methods can be performed for that profile and what the so-called state variables for the profile are. What is in this file should match the services that are offered by the device, but in practice they don't always seem to match.

Eventing

In UPnP there is the concept of so called "state variables". These variables are, as the name says, used for keeping some form of state in UPnP devices and programs. A program can subscribe to state changes: when a state variable is changed, the new state is sent to all programs/devices that have subscribed to the event. A program/device can subscribe to the state variables of a service by subscribing to a URL, which can be found in the URL pointed to by **LOCATION**.

```
<service>
  <serviceType>urn:schemas-upnp-org:service:WANPPPCConnection:1</serviceType>
  <serviceld>urn:upnp-org:serviceld:wanpppc:pppoa</serviceld>
  <controlURL>/upnp/control/wanpppcpppoa</controlURL>
  <eventSubURL>/upnp/event/wanpppcpppoa</eventSubURL>
  <SCPDURL>/WANPPPCConnection.xml</SCPDURL>
</service>
```

The eventing protocol in UPnP is based on [GENA](#).

Presentation

The presentation layer in UPnP refers to the human controllable interface, for example, the webinterface on a router. It is surprising to see that these interfaces often don't match with the functionality that you can use through SOAP. For example, it is often impossible to even see on a router which portmappings are active, let alone change them.

UPnP profiles

Actions and state variables can form a so called 'profile'. The UPnP standardization organizations have standardized a few profiles, which are in widespread use. The most used profiles are:

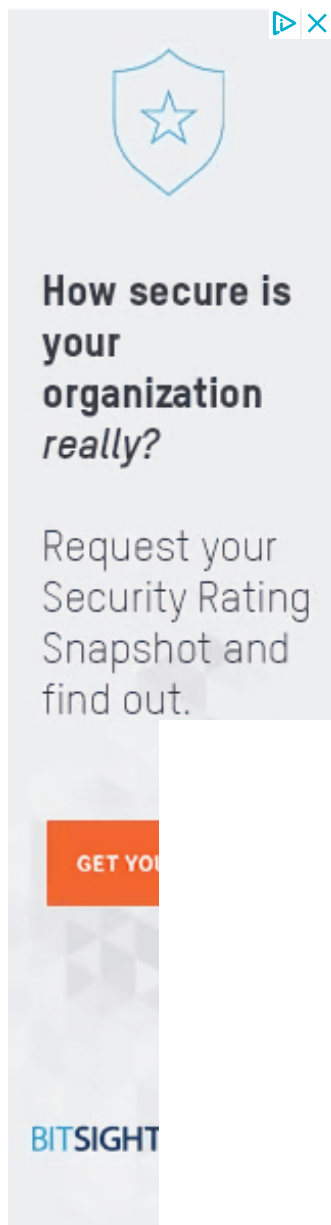
- Internet Gateway Device (IGD)
- Audio/Video (A/V), basis for DLNA

Many profiles have subprofiles, which implement specific behaviour, such as the **WANIPConnection** subprofile in the Internet Gateway Device profile. A device can sometimes implement one (sub)profile multiple times. Top level profiles are usually just containers for several subprofiles.

Implementation caveats

There are a few things you should keep in mind, if you want to implement UPnP functionality for whatever reason.

The most important one is that devices will send responses to M-SEARCH requests to the port that was used for sending the request. You will need to have a program listening on that port. A good solution is to send from UD port 1900 and reuse the socket. There are some incompatibilities between operating systems how you should do this. An extra benefit is that this will also capture a lot of the announcements that are sent regularly over the network.



How secure is your organization really?

Request your Security Rating Snapshot and find out.

GET YOUR SNAPSHOT

BITSIGHT

4. Eventing
5. Presentation

The extra, optional, step is 'addressing'.

Addressing

By default a UPnP-capable device tries to get an IP address through DHCP. If no IP address can be obtained through DHCP an address is chosen in the special link local address range (169.254.0.0/16), similar to what is described in [RFC 3927](#).

Discovery

When a UPnP capable device joins a network and wants to know what UPnP services are available on the network, it sends out a discovery message to the multicast address 239.255.255.250 on port 1900 via the UDP protocol. This message contains a header, similar to a HTTP request. This protocol is sometimes referred to as HTTPU (HTTP over UDP):

```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: ssdp:discover
MX: 10
ST: ssdp:all
```

All other UPnP devices or programs are required to respond to this message by sending a similar message back to the device, using a UDP unicast, announcing which UPnP profiles the device or program implements. An interesting quirk: it is sent back with UDP unicast to the port the device discovery message was sent from. For every profile it implements one message is sent:

```
HTTP/1.1 200 OK
CACHE-CONTROL:max-age=1800
EXT:
LOCATION:http://10.0.0.138:80/IGD.xml
SERVER:SpeedTouch 510 4.0.0.9.0 UPnP/1.0 (DG233B00011961)
ST:urn:schemas-upnp-org:service:WANPPPPConnection:1
USN:uuid:UPnP-SpeedTouch510::urn:schemas-upnp-org:service:WANPPPPConnection:1
```

The above is a slightly edited response that is sent by an Alcatel/Thomson Speedtouch ADSL modem, which implements the **WANPPPPConnection** profile.

At a regular interval UPnP capable devices or programs have to send a message to announce their services. A notification message is more or less the same as a response message to a discovery, but are sent to the UPnP multicast address 239.255.255.250 on port 1900 via UDP and have the ST header replaced by a similar header called NT.

Description

Every profile offers a description of itself and the services it offers and makes this available via XML. The response message from the discovery phase contains a header called **LOCATION** (case insensitive), which is a URL where a file in XML format can be downloaded. This file describes (or rather: should describe) the profile that the device or program implements,