WANIPv6FirewallControl:1 Service

For UPnP Version 1.0
Status: Standardized DCP (SDCP), version 1.00
Date: December 10, 2010
Service Template Version: 2.00

This Standardized DCP has been adopted as a Standardized DCP by the Steering Committee of the UPnP Forum, pursuant to Section 2.1(c)(ii) of the UPnP Forum Membership Agreement. UPnP Forum Members have rights and licenses defined by Section 3 of the UPnP Forum Membership Agreement to use and reproduce the Standardized DCP in UPnP Compliant Devices. All such use is subject to all of the provisions of the UPnP Forum Membership Agreement.

THE UPNP FORUM TAKES NO POSITION AS TO WHETHER ANY INTELLECTUAL PROPERTY RIGHTS EXIST IN THE STANDARDIZED DCPS. THE STANDARDIZED DCPS ARE PROVIDED "AS IS" AND "WITH ALL FAULTS". THE UPNP FORUM MAKES NO WARRANTIES, EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE STANDARDIZED DCPS, INCLUDING BUT NOT LIMITED TO ALL IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE, OF REASONABLE CARE OR WORKMANLIKE EFFORT, OR RESULTS OR OF LACK OF NEGLIGENCE.

Authors[1]	Company
Mark Baugher, Co-editor	Cisco
Mika Saaranen, Chair & Editor	Nokia
Fabrice Fontaine, Co-editor	Orange Labs
Guillaume Habault	Telecom Bretagne

[1] Note: The UPnP Forum in no way guarantees the accuracy or completeness of this author list and in no way implies any rights for or support from those members listed. This list is not the specifications' contributor list that is kept on the UPnP Forum's website.

Contents

List of Tables

List of Figures

1 Overview and Scope

This service definition is compliant with the UPnP Device Architecture version 1.0. It defines a service type referred to herein as *WANIPv6FirewallControl* service.

1.1 Introduction

The *WANIPv6FirewallControl* service is a UPnP service that allows control points to open and manage IPv6 firewall pinholes in the Internet gateway devices regardless of the WAN access type (PPP, DHCP, …). This service provides control points with the following functionality:

- New pinholes can be created to allow external communication to reach an internal local area network. The pinholes have limited duration and they need to be periodically refreshed;
- Control points may check if a certain pinhole is working and allows incoming communication to reach local network. This function is optional for gateway devices;
- Control points may delete the pinholes that they create;

This service deploys access control, as defined in, that restricts operations to duly authorized control points and users.

This service does not provide the following functionality:

- To avoid some potential security issues, it is not possible for control points to retrieve information on any pinholes; For example, a control point can't retrieve information like the external port or the remaining lease duration of any pinhole (even if the CP created this pinhole).
- Control of outbound packet filtering; For example, if the gateway's firewall forbids outbound HTTP connection from the CP to the Internet, the CP can't change this behavior through this service.
- It is not possible for control points to modify the security policy applied by the IGD.

1.2 Notation

- In this document, features are described as Required, Recommended, or Optional as follows:

  The key words “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” in this specification are to be interpreted as described in [RFC 2119].

  In addition, the following keywords are used in this specification:

  PROHIBITED – The definition or behavior is an absolute prohibition of this specification. Opposite of REQUIRED.

  CONDITIONALLY REQUIRED – The definition or behavior depends on a condition. If the specified condition is met, then the definition or behavior is REQUIRED, otherwise it is PROHIBITED.

  CONDITIONALLY OPTIONAL – The definition or behavior depends on a condition. If the specified condition is met, then the definition or behavior is OPTIONAL, otherwise it is PROHIBITED.

  These keywords are thus capitalized when used to unambiguously specify requirements over protocol and application features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

- Strings that are to be taken literally are enclosed in “double quotes”.
- Words that are emphasized are printed in *italic*.

- Keywords that are defined by the UPnP Working Committee are printed using the *forum* character style.
- Keywords that are defined by the UPnP Device Architecture are printed using the **arch** character style.
- A double colon delimiter, "::", signifies a hierarchical parent-child (parent::child) relationship between the two objects separated by the double colon. This delimiter is used in multiple contexts, for example: Service::Action(), Action()::Argument, parentProperty::childProperty.

1.2.1 Data Types

This specification uses data type definitions from two different sources. The UPnP Device Architecture defined data types are used to define state variable and action argument data types [DEVICE]. The XML Schema namespace is used to define property data types [XML SCHEMA-2].

For UPnP Device Architecture defined Boolean data types, it is strongly RECOMMENDED to use the value "**0**" for false, and the value "**1**" for true. The values "**true**", "**yes**", "**false**", or "**no**" MAY also be used but are NOT RECOMMENDED. The values "**yes**" and "**no**" are deprecated and MUST NOT be sent out by devices but MUST be accepted on input.

For XML Schema defined Boolean data types, it is strongly RECOMMENDED to use the value "*0*" for false, and the value "*1*" for true. The values "*true*", "*yes*", "*false*", or "*no*" MAY also be used but are NOT RECOMMENDED. The values "*yes*" and "*no*" are deprecated and MUST NOT be sent out by devices but MUST be accepted on input.

1.3 Vendor-defined Extensions

Whenever vendors create additional vendor-defined state variables, actions or properties, their assigned names and XML representation MUST follow the naming conventions and XML rules as specified in [DEVICE], Section 2.5, "Description: Non-standard vendor extensions".

1.4 References

1.4.1 Normative References

This section lists the normative references used in this specification and includes the tag inside square brackets that is used for each such reference:

[IGD2] – UPnP *InternetGatewayDevice:2*, version 1.00, UPnP Forum, December 10, 2010.
Available at http://upnp.org/specs/gw/UPnP-gw-InternetGatewayDevice-v2-Device.pdf.

[WANDevice] – UPnP *WANDevice:2*, version 1.0, UPnP Forum, September 10, 2010.
Available at http://upnp.org/specs/gw/UPnP-gw-WANDevice-v2-Device.pdf.

[WANConnectionDevice] – UPnP *WANConnectionDevice:2*, version 1.00, UPnP Forum, September 10, 2010.
Available at: http://upnp.org/specs/gw/UPnP-gw-WANConnectionDevice-v2-Device.pdf.

[WANIPConnection] – UPnP *WANIPConnection:2*, version 1.00, UPnP Forum, September 10, 2010.
Available at: http://upnp.org/specs/gw/UPnP-gw-WANIPConnection-v2-Service.pdf.

[DEVICE] – UPnP Device Architecture, version 1.0, UPnP Forum, June 8, 2000.
Available at: http://upnp.org/specs/arch/UPnPDA10_20000613.pdf.
Latest version available at: http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0.pdf.

[RFC6092] –IETF RFC 6092, Recommended Simple Security Capabilities in Customer Premises Equipment (CPE) for Providing Residential IPv6 Internet Service, J. Woodyatt, January 2011
Available at: http://tools.ietf.org/html/rfc6092.

[ISO 8601] – Data elements and interchange formats – Information interchange -- Representation of dates and times, International Standards Organization, December 21, 2000.
Available at: ISO 8601:2000.

[RFC 2119] – IETF RFC 2119, Key words for use in RFCs to Indicate Requirement Levels, S. Bradner, March 1997.
Available at: http://tools.ietf.org/html/rfc2119.

[RFC 4291] – IETF RFC 4291, IP Version 6 Addressing Architecture, R. Hinden, S. Deering, February 2006.
Available at: http://tools.ietf.org/html/rfc4291.

[RFC 4890] – IETF RFC 4890, Recommendations for Filtering ICMPv6 Messages in Firewalls, E. Davies, J. Mohacsi, May 2007.
Available at: http://tools.ietf.org/html/rfc4890.

[RFC 3986] – IETF RFC 3986, Uniform Resource Identifier (URI): Generic Syntax, T. Berners-Lee, R. Fielding, L.Masinter, January 2005.
Available at: http://tools.ietf.org/html/rfc3986.

[RFC 3339] – IETF RFC 3339, Date and Time on the Internet: Timestamps, G. Klyne, Clearswift Corporation, C. Newman, Sun Microsystems, July 2002.
Available at: http://tools.ietf.org/html/rfc3339.

[XML] – Extensible Markup Language (XML) 1.0 (Third Edition), François Yergeau, Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, eds., W3C Recommendation, February 4, 2004.
Available at: http://www.w3.org/TR/2004/REC-xml-20040204.

[XML SCHEMA-2] – XML Schema Part 2: Data Types, Second Edition, Paul V. Biron, Ashok Malhotra, W3C Recommendation, 28 October 2004.
Available at: http://www.w3.org/TR/2004/REC-xmlschema-2-20041028.

1.4.2 Informative References

This section lists the informative references that are provided as information in helping understand this specification:

[CB] W.R. Cheswick and Bellovin, S.M., Firewalls and Internet Security, Addison-Wesley, 1994.

2 Service Modeling Definitions (Normative)

2.1 Service Type

The following service type identifies a service that is compliant with this specification:

urn:schemas-upnp-org:service:*WANIPv6FirewallControl:1*

WANIPv6FirewallControl service is used herein to refer to this service type.

2.2 Terms and Abbreviations

2.2.1 Abbreviations

Table 2-1: Abbreviations

Abbreviation	Description
CP	Control Point
DCP	Device Control Protocol
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Service
GUA	Global Unicast Address
HTTP	HyperText Transfer Protocol
IANA	Internet Assigned Numbers Authority
ICMPv6	Internet Control Message Protocol version 6
IGD	Internet Gateway Device
IPv6	Internet Protocol version 6
LAN	Local Area Network
MTU	Maximum Transmission Unit
NAT	Network Address Translation
P2P	Peer to Peer
PMP	Port Mapping Protocol
PPP	Point to Point Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
WAN	Wide Area Network

2.2.2 Terms

2.2.2.1 Firewall

A firewall inspects network traffic passing through it, and denies or permits the passage based on a set of rules.

A firewall has an "external" interface and an "internal" interface (for example a LAN interface).

In most firewalls, the external interface will be connected to the WAN and the internal interface will be connected to the LAN. However, some firewalls may have both interface connected to two different LANs.

2.2.2.2 Pinhole

A pinhole is the opening of a firewall packet filter to allow unsolicited packets originating from the external interface of the firewall to the corresponding internal address. Unlike IPv4 network address translation, the external address is not re-written to a mapped internal address but are in one-to-one correspondence. In IPv6, this address is typically a globally-routable IPv6 Global Unicast Address.

2.2.2.3 Inbound traffic

An inbound traffic is a traffic going from the external to the internal interface of the firewall. For example, a remote host in the WAN sending traffic to one of the internal client in the IGD's local network.

2.2.2.4 Outbound traffic

An outbound traffic is a traffic going from the internal to the external interface of the firewall. For example, one of the internal client in the IGD's local network sending traffic to a remote host in the WAN.

2.2.2.5 Inbound pinhole

An inbound pinhole is a rule created in the firewall to allow inbound initiated traffic to pass through the firewall.

If an inbound pinhole is created and an inbound traffic is initiated, it is REQUIRED that all the incoming and outgoing traffic will be allowed for the duration of this traffic session.

2.2.2.6 Outbound pinhole

An outbound pinhole is a rule created in the firewall to allow outbound initiated traffic to pass through the firewall.

If an outbound pinhole is created and an outbound traffic is initiated, it is REQUIRED that all the incoming and outgoing traffic will be allowed for the duration of this traffic session.

Note: In most gateways and routers, the default behavior is to create, automatically, an outbound pinhole when an outbound traffic is initiated. This kind of pinhole is called an "automatic pinhole" in this document.

2.2.2.7 Traffic session

A traffic session is defined as an exchange of outbound and inbound traffic between an internal-client interface that is identified by a transport address and a remote host for a given duration.

2.3 *WANIPv6FirewallControl* Service Architecture

This service controls the firewall to create pinholes for incoming traffic and also retrieves information from the firewall. More specifically the main functionalities provided by this service are:

1. Create pinholes with action *AddPinhole()*. Pinholes exist until their lease time expires and then are automatically deleted. If a longer duration is needed, then the control point needs to use the *UpdatePinhole()* action or the *AddPinhole()* action again. CP's identify pinholes by the *UniqueID* value that they retrieve when creating a pinhole.
2. Update lease time of a pinhole by *UpdatePinhole()* or *AddPinhole()*. The maximum time that a pinhole can exist without updating is one day. If a pinhole is not updated with a new lease time, then the pinhole is automatically deleted after it expires.
3. *DeletePinhole()* allows removing pinholes.
4. *GetOutboundPinholeTimeout()* action allows retrieving default value after which automatically created pinholes expire.

5. *GetFirewallStatus()* allows control points to know if the firewall is enabled and if pinhole can be created through UPnP.
6. *GetPinholePackets()* allows control points to get the total number of IP packets which have been going through a certain pinhole.
7. *CheckPinholeWorking()* is an optional action that allows checking if a certain pinhole allows traffic to pass through the firewall.

2.4 State Variables

The state variables include two state variables describing the status of a firewall, ie whether or not pinholes may be created and does the firewall is enabled. The other type of variables are arguments, which are distinct from state variables but that could be used to return information to the CP such as how long automatically created pinholes exist.

Note: For first-time reader, it may be more insightful to read the theory of operations first and then the action definitions before reading the state variable definitions.

2.4.1 State Variable Overview

Table 2-2: State Variables

Variable Name	R/O[1]	Data Type	Reference
FirewallEnabled	*R*	**Boolean**	See Section 2.4.2
InboundPinholeAllowed	*R*	**Boolean**	See Section 2.4.3
A_ARG_TYPE_OutboundPinholeTimeout	*O*	**ui4**	See Section 2.4.4
A_ARG_TYPE_IPv6Address	*R*	**string** (IPv6 address in hexadecimal or DNS name)	See Section 2.4.5
A_ARG_TYPE_Port	*R*	**ui2**	See Section 2.4.6
A_ARG_TYPE_Protocol	*R*	**ui2**	See Section 2.4.7
A_ARG_TYPE_LeaseTime	*R*	**ui4**	See Section 2.4.8
A_ARG_TYPE_UniqueID	*R*	**ui2**	See Section 2.4.9
A_ARG_TYPE_PinholePackets	*R*	**ui4**	See Section 2.4.10
A_ARG_TYPE_Boolean	*O*	**Boolean**	See Section 2.4.11
Non standard state variables implemented by an UPnP vendor go here	*X*	*TBD*	*TBD*

[1] *R* = REQUIRED, *O* = OPTIONAL, *CR* = CONDITIONALLY REQUIRED, *CO* = CONDITIONALLY OPTIONAL, *X* = Non-standard, add *-D* when deprecated (e.g., *R-D*, *O-D*).

2.4.2 *FirewallEnabled*

The type of this variable is **boolean,** and it is used to notify of changes if the firewall is enabled.

If this variable is set to "**0**" (false), the firewall is not enabled and so all inbound and outbound traffic is allowed to go though the IGD. In other words, UPnP CPs don't have to create pinholes to allow inbound connections.

If this variable is set to "**1**" (true), the firewall is enabled and so UPnP CPs SHOULD check *InboundPinholeAllowed* state variable to know if the IGD allows UPnP CPs to create inbound pinholes in the firewall.

2.4.3 *InboundPinholeAllowed*

The type of this variable is **boolean,** and it is used to notify of changes if UPnP CPs are allowed to create pinholes.

If this variable is set to "**1**" (true), the IGD allows UPnP CPs to create pinholes to allow inbound traffic through the *AddPinhole()* action.

If this variable is set to "**0**" (false), the IGD doesn't allow UPnP CPs to create pinholes to allow inbound traffic through the *AddPinhole()* action. Moreover, if this variable is set to "**0**" (false), all pinholes created previously by any UPnP Control Points will be deleted by the IGD, even if their lease time has not expired yet.

2.4.4 *A_ARG_TYPE_OutboundPinholeTimeout*

Table 2-3: allowedValueRange for the *A_ARG_TYPE_OutboundPinholeTimeout* state variable

	Value	R/O
minimum	*Vendor-defined* (RECOMMENDED value is 120)	*R*
maximum	*Vendor-defined*	*R*
default	*N/A*	N/A

This variable is of type **ui2** and determines the lifetime in seconds of an inbound "automatic" firewall pinhole created by an outbound traffic initiation.

This document RECOMMENDS 120 seconds as the minimum value (as defined in [REC-12] of [RFC6092]).

2.4.5 *A_ARG_TYPE_IPv6Address*

This variable is of type **string** and represents the source or the destination of inbound IPv6 packets. This variable can be defined as either literal presentation of IPv6 address as defined in [RFC 4291], as domain name defined in [RFC 3986] or as a wildcard (an empty string).

2.4.6 *A_ARG_TYPE_Port*

This variable is of type **ui2** and represents the source or destination port of the transport protocol used in pinhole. Value "0" is used to describe any value. Value "0" is the wildcard value for this variable.

2.4.7 *A_ARG_TYPE_Protocol*

This variable is of type **ui2** and its value enumerates the protocol of the pinhole. The specific values MUST be according to the Internet Assigned Numbers Authority [IANA protocol]. Values below 256 are reserved for IANA defined usage, values greater than 255 are left undefined at this moment, except, 65535 which is reserved for presenting any possible protocol. 65535 is the wildcard value for this variable.

2.4.8 *A_ARG_TYPE_LeaseTime*

Table 2-4: allowedValueRange for the *A_ARG_TYPE_LeaseTime* state variable

	Value	R/O
minimum	*1*	*R*

	Value	R/O
maximum	86400	R
default	N/A	N/A

This variable determines the lifetime in seconds of a pinhole and indicates the duration after which a pinhole will be removed, unless a control point refreshes it.

A lease time greater than or equal to 3600 is RECOMMENDED by this service.

2.4.9 A_ARG_TYPE_UniqueID

This variable is of type **ui2** and it gives unique identifier of the pinhole corresponding to the address, port and protocol.

2.4.10 A_ARG_TYPE_PinholePackets

This variable is of type **ui4** represents the cumulative counter for total number of IP packets which have been going through a specified inbound pinhole. The count rolls over to 0 after it reaching the maximum value (2^32) –1.

2.4.11 A_ARG_TYPE_Boolean

This **boolean** type argument is used to carry **boolean** type information as arguments.

2.5 Eventing and Moderation

Table 2-5: Eventing and Moderation

Variable Name	Evented	Moderated	Criteria
FirewallEnabled	YES	NO	N/A
InboundPinholeAllowed	YES	NO	N/A

2.5.1 Eventing of FirewallEnabled

This variable is evented everytime the state variable is updated. There is no moderation defined.

2.5.2 Eventing of InboundPinholeAllowed

This variable is evented everytime the state variable is updated. There is no moderation defined.

2.5.3 Relationships among State Variables

The relationships between firewall state variables are:

- FirewallEnabled value impacts to other state variables by controlling whether firewall as such is working or it can be controlled;
- InboundPinholeAllowed value impacts to other state variables by controlling whether firewall as such can be controlled by UPnP control points.

2.6 Actions

Table 2-6: Actions

Name	Device R/O[1]	Control Point R/O[2]
GetFirewallStatus()	R	O

Name	Device R/O[1]	Control Point R/O[2]
GetOutboundPinholeTimeout()	O	O
AddPinhole()	R	R
UpdatePinhole()	R	R
DeletePinhole()	R	O
GetPinholePackets()	R	O
CheckPinholeWorking()	O	O
Non-standard actions implemented by an UPnP vendor go here.	*X*	*X*

[1] For a device this column indicates whether the action MUST be implemented or not, where *R* = REQUIRED, *O* = OPTIONAL, *CR* = CONDITIONALLY REQUIRED, *CO* = CONDITIONALLY OPTIONAL, *X* = Non-standard, add *-D* when deprecated (e.g., *R-D*, *O-D*).

[2] For a control point this column indicates whether a control point MUST be capable of invoking this action, where *R* = REQUIRED, *O* = OPTIONAL, *CR* = CONDITIONALLY REQUIRED, *CO* = CONDITIONALLY OPTIONAL, *X* = Non-standard, add *-D* when deprecated (e.g., *R-D*, *O-D*).

2.6.1 *GetFirewallStatus()*

This action is to detect if the firewall is active and allows creating pinholes by control points.

2.6.1.1 Arguments

Table 2-7: Arguments for *GetFirewallStatus()*

Argument	Direction	relatedStateVariable
FirewallEnabled	*OUT*	*FirewallEnabled*
InboundPinholeAllowed	*OUT*	*InboundPinholeAllowed*

2.6.1.2 Argument Descriptions

FirewallEnabled has type **boolean** and informs if the firewall is active.

InboundPinholeAllowed has type **boolean** and it informs if inbound pinhole can be created through UPnP.

2.6.1.3 Service Requirements

Before processing the action request, the device MUST apply the chosen security policy, and authenticate and authorize the control point as required by the security policy. If the control point does not have the necessary permission to perform the action with the requested parameters, the device MUST return the 606 *Action not authorized* error code (defined in [DEVICE]). [IGD2] RECOMMENDS access control requirements and authentication levels to be applied by *default* for this action. However, devices MAY choose a different security policy.

2.6.1.4 Control Point Requirements When Calling The Action

Before invoking this action, a control point SHOULD verify that it has sufficient permission.

2.6.1.5 Dependency on Device State

None.

2.6.1.6 Effect on Device State

None.

2.6.1.7 Errors

Table 2-8: Error Codes for *GetFirewallStatus()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
606	*Action not authorized*	The action requested REQUIRES authorization and the sender was not authorized.

2.6.2 *GetOutboundPinholeTimeout()*

This action returns the outbound pinhole timeout for the "automatic pinhole" defined by arguments. The returned value MAY be specific to the *Protocol*, *RemoteHost*, *RemotePort*, *InternalClient* and *InternalPort*, but this behavior depends on the implementation of the firewall. For instance, time-out value can be different for well-known ports(<1024) than higher ports for UDP protocol as defined in [REC-14] and [REC-15] of [RFC6092]. By using a smaller timeout for well-known ports, the IGD vendor can:

- better protect the internal clients against attacks;
- facilitate the operation of IANA-registered service assigned to the port in question.

2.6.2.1 Arguments

Table 2-9: Arguments for *GetOutboundPinholeTimeout()*

Argument	Direction	relatedStateVariable
RemoteHost	*IN*	*A_ARG_TYPE_IPv6Address*
RemotePort	*IN*	*A_ARG_TYPE_Port*
InternalClient	*IN*	*A_ARG_TYPE_IPv6Address*
InternalPort	*IN*	*A_ARG_TYPE_Port*
Protocol	*IN*	*A_ARG_TYPE_Protocol*
OutboundPinholeTimeout	*OUT*	*A_ARG_TYPE_OutboundPinholeTimeout*

2.6.2.2 Argument descriptions

RemoteHost is **string** type variable that describes source address for the outbound pinhole. This can be wildcarded.

RemotePort is **ui2** type variable that describes source port for the outbound pinhole. This can be wildcarded.

InternalClient is **string** type variable that describes destination address for the outbound pinhole. This can be wildcarded.

InternalPort is **ui2** type variable that describes destination port for the outbound pinhole. This can be wildcarded.

Protocol is **ui2** type variable that describes the protocol for the outbound pinhole. This can be wildcarded.

OutboundPinholeTimeout is **ui4** type variable that defines outbound pinhole timeout for "'automatic pinhole".

2.6.2.3 Service Requirements

Before processing the action request, the device MUST apply the chosen security policy, and authenticate and authorize the control point as required by the security policy. If the control point does not have the necessary permission to perform the action with the requested parameters, the device MUST return the 606 *Action not authorized* error code (defined in [DEVICE]). [IGD2] RECOMMENDS access control requirements and authentication levels to be applied by *default* for this action. However, devices MAY choose a different security policy.

If this action has been implemented, it is REQUIRED to return values for UDP and UDPLite. Support for other protocols is OPTIONAL. If the requested protocol is not supported, device MUST return error 705 *ProtocolNotSupported*.

If wildcard values are used then it is REQUIRED that shortest timeout value is returned.

2.6.2.4 Control Point Requirements When Calling The Action

Before invoking this action, a control point SHOULD verify that it has sufficient permission.

2.6.2.5 Dependency on Device State

FirewallEnabled MUST be true indicating that firewall is active.

2.6.2.6 Effect on Device State

None.

2.6.2.7 Errors

Table 2-10: Error Codes for *GetOutboundPinholeTimeout()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
606	*Action not authorized*	The action requested REQUIRES authorization and the sender was not authorized.
702	*FirewallDisabled*	Firewall is disabled and this action is disabled.
705	*ProtocolNotSupported*	Specified protocol is not supported by this action.

2.6.3 *AddPinhole()*

This action allows a control point to create a new pinhole that allows incoming traffic to pass through firewall.

This action can also be used by a control point to extend the lease time of an existing pinhole.

2.6.3.1 Arguments

Table 2-11: Arguments for *AddPinhole()*

Argument	Direction	relatedStateVariable
RemoteHost	*IN*	*A_ARG_TYPE_IPv6Address*

Argument	Direction	relatedStateVariable
RemotePort	*IN*	*A_ARG_TYPE_Port*
InternalClient	*IN*	*A_ARG_TYPE_IPv6Address*
InternalPort	*IN*	*A_ARG_TYPE_Port*
Protocol	*IN*	*A_ARG_TYPE_Protocol*
LeaseTime	*IN*	*A_ARG_TYPE_LeaseTime*
UniqueID	*OUT*	*A_ARG_TYPE_UniqueID*

2.6.3.2 Argument Descriptions

RemoteHost is **string** type variable that describes source address for the pinhole. This can be wildcarded.

RemotePort is **ui2** type variable that describes source port for the pinhole. This can be wildcarded.

InternalClient is **string** type variable that describes destination address for the pinhole. This can not be wildcarded.

InternalPort is **ui2** type variable that describes destination port for the pinhole. This can be wildcarded. This service RECOMMENDS the wildcard support however a device is allowed to return an error code if it does not support it.

Protocol is **ui2** type variable that describes the protocol that uses this pinhole. This can be wildcarded. This service RECOMMENDS the wildcard support however a device is allowed to return an error code if it does not support it.

LeaseTime is **ui4** type variable that defines expiration time of the pinhole.

UniqueID is **ui2** type variable that identifies the firewall pinhole.

2.6.3.3 Service Requirements

Before processing the action request, the device MUST apply the chosen security policy, and authenticate and authorize the control point as required by the security policy. If the control point does not have the necessary permission to perform the action with the requested parameters, the device MUST return the 606 *Action not authorized* error code (defined in [DEVICE]).

[IGD2] RECOMMENDS access control requirements and authentication levels to be applied by *default* for this action. However, devices MAY choose a different security policy.

In particular, [IGD2] RECOMMENDS that unauthenticated and unauthorized control points are only allowed to invoke this action with:

- *InternalPort* value greater than or equal to 1024,
- *InternalClient* value equals to the control point's IP address.

It is REQUIRED that *InternalClient* cannot be one of IPv6 addresses used by the gateway.

In cases where the *RemoteHost*, *RemotePort*, *InternalPort*, *InternalClient* and *Protocol* are the same than an existing pinhole, but *LeaseTime* is different, the device MUST extend the existing pinhole's lease time and return the *UniqueID* of the existing pinhole.

2.6.3.4 Control Point Requirements When Calling The Action

Before invoking this action, a control point SHOULD verify that it has sufficient permission.

Control points that have not been authenticated and authorized as defined in [IGD2] SHOULD use their IPv6 GUA when calling this action.

2.6.3.5 Dependency on Device State

FirewallEnabled MUST be true indicating that firewall is active.

InboundPinholeAllowed MUST be true indicating that UPnP CPs can create inbound pinhole.

Also, CP needs to have appropriate access rights to complete this action.

2.6.3.6 Effect on Device State

A new firewall pinhole is created.

2.6.3.7 Errors

Table 2-12: Error Codes for *AddPinhole()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
606	*Action not authorized*	The action requested REQUIRES authorization and the sender was not authorized. This error can be returned e.g. if a control point is not authorized for the operation because of its IP address or because it tries to use well-known ports.
701	*PinholeSpaceExhausted*	Firewall cannot accommodate more pinholes at this moment.
702	*FirewallDisabled*	Firewall is disabled and this action is disabled.
703	*InboundPinholeNotAllowed*	Creation of inbound pinholes by UPnP CPs are not allowed and this action is disabled.
705	*ProtocolNotSupported*	Specified protocol is not supported by this action.
706	*InternalPortWildcardingNotAllowed*	Gateway does not allow wildcarding *InternalPort* value.
707	*ProtocolWildcardingNotAllowed*	Gateway does not allow wildcarding *Protocol* value.
708	*WildCardNotPermittedInSrcIP*	The source IP address cannot be wild-carded (*InternalClient* equals to an empty string).

2.6.4 *UpdatePinhole()*

This action updates a pinhole's lease time.

2.6.4.1 Arguments

Table 2-13: Arguments for *UpdatePinhole()*

Argument	Direction	relatedStateVariable
UniqueID	*IN*	*A_ARG_TYPE_UniqueID*
NewLeaseTime	*IN*	*A_ARG_TYPE_LeaseTime*

2.6.4.2 Argument Descriptions

UniqueID argument gives unique identifier assigned earlier by the gateway. Type is **ui2**.

NewLeaseTime defines how long pinhole will exist. Type is **ui4**.

2.6.4.3 Service Requirements

Before processing the action request, the device MUST apply the chosen security policy, and authenticate and authorize the control point as required by the security policy. If the control point does not have the necessary permission to perform the action with the requested parameters, the device MUST return the 606 *Action not authorized* error code (defined in [DEVICE]). [IGD2] RECOMMENDS access control requirements and authentication levels to be applied by *default* for this action. However, devices MAY choose a different security policy.

In particular, [IGD2] RECOMMENDS that unauthenticated and unauthorized control points are only allowed to update pinholes which have:

- *InternalPort* value greater than or equal to 1024,
- *InternalClient* value equals to the control point's IP address.

2.6.4.4 Control Point Requirements When Calling The Action

Before invoking this action, a control point SHOULD verify that it has sufficient permission.

Control points that have not been authenticated and authorized as defined in [IGD2] SHOULD use their IPv6 GUA when updating a firewall pinhole.

2.6.4.5 Dependency on Device State

FirewallEnabled MUST be true indicating that firewall is active.

InboundPinholeAllowed MUST be true indicating that UPnP CPs can create and update inbound pinhole.

CP MUST have sufficient access rights to update pinhole.

2.6.4.6 Effect on Device State

The effect is that the pinhole's lease time is extended.

2.6.4.7 Errors

Table 2-14: Error Codes for *UpdatePinhole()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
606	*Action not authorized*	The action requested REQUIRES authorization and the sender was not authorized. This error can be returned e.g. if a control point is not authorized for the operation because of its IP address or because it tries to use well-known ports.
702	*FirewallDisabled*	Firewall is disabled and this action is disabled.
703	*InboundPinholeNotAllowed*	Creation of inbound pinholes by UPnP CPs are not allowed and this action is disabled.
704	*NoSuchEntry*	There is no pinhole with the specified *UniqueID*.

2.6.5 DeletePinhole()

This action removes a pinhole.

2.6.5.1 Arguments

Table 2-15: Arguments for DeletePinhole()

Argument	Direction	relatedStateVariable
UniqueID	IN	A_ARG_TYPE_UniqueID

2.6.5.2 Argument Descriptions

This argument gives unique identifier assigned earlier by the gateway. Type is **ui2**.

2.6.5.3 Service Requirements

Before processing the action request, the device MUST apply the chosen security policy, and authenticate and authorize the control point as required by the security policy. If the control point does not have the necessary permission to perform the action with the requested parameters, the device MUST return the 606 *Action not authorized* error code (defined in [DEVICE]).

[IGD2] RECOMMENDS access control requirements and authentication levels to be applied by *default* for this action. However, devices MAY choose a different security policy.

In particular, [IGD2] RECOMMENDS that unauthenticated and unauthorized control points are only allowed to delete pinholes which have:

- *InternalPort* value greater than or equal to 1024,
- *InternalClient* value equals to the control point's IP address.

2.6.5.4 Control Point Requirements When Calling The Action

Before invoking this action, a control point SHOULD verify that it has sufficient permission.

Control points that have not been authenticated and authorized as defined in [IGD2] SHOULD use their IPv6 GUA when deleting a firewall pinhole.

2.6.5.5 Dependency on Device State

FirewallEnabled MUST be true indicating that firewall is active.

InboundPinholeAllowed MUST be true indicating that UPnP CPs can create and delete inbound pinhole.

CP MUST have sufficient access rights to delete pinhole.

2.6.5.6 Effect on Device State

The effect is that specified pinhole is deleted and traffic can no more pass the firewall.

2.6.5.7 Errors

Table 2-16: Error Codes for DeletePinhole()

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.

ErrorCode	errorDescription	Description
606	Action not authorized	The action requested REQUIRES authorization and the sender was not authorized. This error can be returned e.g. if a control point is not authorized for the operation because of its IP address or because it tries to use well-known ports.
702	FirewallDisabled	Firewall is disabled and this action is disabled.
703	InboundPinholeNotAllowed	Creation of inbound pinholes by UPnP CPs are not allowed and this action is disabled.
704	NoSuchEntry	There is no pinhole with the specified UniqueID.

2.6.6 GetPinholePackets()

This action allows a control point to get the total number of IP packets which have been going through the specified pinhole.

This action could be used by a CP to know if a certain pinhole is "working". In fact, if the PinholePackets value is greater than zero, it means that the pinhole had worked, at least once. Moreover, if the CP retrieves the PinholePackets value at a later time, and the value has increased, the CP can suppose that the pinhole is still working.

Note: The major limitation of this action is that outbound traffic is needed, otherwise this action will always return zero. Moreover, an other limitation is that some devices specify that all established and related network packets should pass automatically the firewall. As a result, with those devices, the PinholePackets value will not increase even if there is packets going through the device as only the first packet will go through the pinhole.

2.6.6.1 Arguments

Table 2-17: Arguments for GetPinholePackets()

Argument	Direction	relatedStateVariable
UniqueID	IN	A_ARG_TYPE_UniqueID
PinholePackets	OUT	A_ARG_TYPE_PinholePackets

2.6.6.2 Argument Descriptions

UniqueID is unique identifier for a pinhole returned by AddPinhole() action. Type is **ui2**.

PinholePackets is a **ui4** type variable describing how many IP packets have been going through the specified pinhole.

2.6.6.3 Service Requirements

Before processing the action request, the device MUST apply the chosen security policy, and authenticate and authorize the control point as required by the security policy. If the control point does not have the necessary permission to perform the action with the requested parameters, the device MUST return the 606 Action not authorized error code (defined in [DEVICE]). [IGD2] RECOMMENDS access control requirements and authentication levels to be applied by *default* for this action. However, devices MAY choose a different security policy.

In particular, [IGD2] RECOMMENDS that unauthenticated and unauthorized control points are only allowed to check pinholes which have:

- InternalPort value greater than or equal to 1024,

- *InternalClient* value equals to the control point's IP address.

2.6.6.4 Control Point Requirements When Calling The Action

Before invoking this action, a control point SHOULD verify that it has sufficient permission.

Control points that have not been authenticated and authorized as defined in [IGD2] SHOULD use their IPv6 GUA when getting the packets for a firewall pinhole.

2.6.6.5 Dependency on Device State

FirewallEnabled MUST be true indicating that firewall is active.

InboundPinholeAllowed MUST be true indicating that UPnP CPs can create pinhole.

2.6.6.6 Effect on Device State

None.

2.6.6.7 Errors

Table 2-18: Error Codes for *GetPinholePackets()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
606	*Action not authorized*	The action requested REQUIRES authorization and the sender was not authorized.
702	*FirewallDisabled*	Firewall is disabled and this action is disabled.
703	*InboundPinholeNot Allowed*	Creation of inbound pinholes by UPnP CPs are not allowed and this action is disabled.
704	*NoSuchEntry*	There is no pinhole with the specified *UniqueID*.

2.6.7 *CheckPinholeWorking()*

This action allows a control point to verify if a certain pinhole allows traffic to pass through the firewall. It is possible that other pinholes created by non-UPnP means effectively render specified pinhole obsolete and no traffic can pass through. Logic and implementation of this action are left to implementers. This action returns a **boolean** value indicating if traffic can pass through the firewall.

Note: It could be very difficult to know if a pinhole is working before any traffic is actually received from the remote host. As a result, vendors MAY decide to implement this function by returning true if some traffic was received and processed through the specified pinhole. As a result, the device MAY return the 709 *NoTrafficReceived* error code if no traffic corresponding to the specified pinhole was received.

2.6.7.1 Arguments

Table 2-19: Arguments for *CheckPinholeWorking()*

Argument	Direction	relatedStateVariable
UniqueID	*IN*	*A_ARG_TYPE_UniqueID*
IsWorking	*OUT*	*A_ARG_TYPE_Boolean*

2.6.7.2 Argument Descriptions

UniqueID is unique identifier for a pinhole returned by *AddPinhole()* action. Type is **ui2**.

IsWorking is a **boolean** type variable describing if specified pinhole will allow traffic to pass.

2.6.7.3 Service Requirements

Before processing the action request, the device MUST apply the chosen security policy, and authenticate and authorize the control point as required by the security policy. If the control point does not have the necessary permission to perform the action with the requested parameters, the device MUST return the 606 *Action not authorized* error code (defined in [DEVICE]).

[IGD2] RECOMMENDS access control requirements and authentication levels to be applied by *default* for this action. However, devices MAY choose a different security policy.

In particular, [IGD2] RECOMMENDS that unauthenticated and unauthorized control points are only allowed to check pinholes which have:

- *InternalPort* value greater than or equal to 1024,
- *InternalClient* value equals to the control point's IP address.

2.6.7.4 Control Point Requirements When Calling The Action

Before invoking this action, a control point SHOULD verify that it has sufficient permission.

Control points that have not been authenticated and authorized as defined in [IGD2] SHOULD use their IPv6 GUA when checking a firewall pinhole.

2.6.7.5 Dependency on Device State

FirewallEnabled MUST be true indicating that firewall is active.

InboundPinholeAllowed MUST be true indicating that UPnP CPs can create pinhole.

2.6.7.6 Effect on Device State

None.

2.6.7.7 Errors

Table 2-20: Error Codes for *CheckPinholeWorking()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
606	*Action not authorized*	The action requested REQUIRES authorization and the sender was not authorized.
702	*FirewallDisabled*	Firewall is disabled and this action is disabled.
703	*InboundPinholeNotAllowed*	Creation of inbound pinholes by UPnP CPs are not allowed and this action is disabled.
704	*NoSuchEntry*	There is no pinhole with the specified *UniqueID*.
709	*NoTrafficReceived*	No traffic corresponding to this pinhole has been received by the gateway.

2.6.8 Relationships Between Actions

The relationship between actions are:

AddPinhole() creates pinholes that allow incoming traffic through the firewall.

UpdatePinhole() allows extending life time of a pinhole with *UniqueID* returned by *AddPinhole()*.

Pinholes are automatically deleted:

- after their *LeaseTime* expires;
- by *DeletePinhole()* action.

2.6.9 Error Code Summary

The following table lists error codes common to actions for this service type. If an action results in multiple errors, the most specific error should be returned.

Table 2-21: Error Code Summary

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
606	*Action not authorized*	The action requested REQUIRES authorization and the sender was not authorized. This error can be returned e.g. if a control point is not authorized for the operation because of its IP address or because it tries to use well-known ports.
700		Reserved for future extensions.
701	*PinholeSpaceExhausted*	Firewall cannot accommodate more pinholes at this moment.
702	*FirewallDisabled*	Firewall is disabled and this action is disabled.
703	*InboundPinholeNotAllowed*	Creation of inbound pinholes by UPnP CPs are not allowed and this action is disabled.
704	*NoSuchEntry*	There is no pinhole with the specified *UniqueID*.
705	*ProtocolNotSupported*	Specified protocol is not supported by this action.
706	*InternalPortWildcardingNotAllowed*	Gateway does not allow wildcarding *InternalPort* value.
707	*ProtocolWildcardingNotAllowed*	Gateway does not allow wildcarding *Protocol* value.
708	*WildCardNotPermittedInSrcIP*	The source IP address cannot be wild-carded (*InternalClient* equals to an empty string).
709	*NoTrafficReceived*	No traffic corresponding to this pinhole has been received by the gateway.

Note: 800-899 Error Codes are not permitted for standard actions. See UPnP Device Architecture section on Control for more details.

2.7 Service Behavioral Model

2.7.1 Operation requirements

It is RECOMMENDED that, at least, the following ICMPv6 traffic related to a traffic session is allowed to go through the firewall as specified in section [4.3.1] of [RFC 4890] and [REC-18], [REC-36], [REC-41], [REC-45] of [RFC6092]:

- Destination Unreachable (Type 1) – All codes
- Packet Too Big (Type 2)

It is RECOMMENDED that, at least, the following ICMPv6 traffic related to a traffic session is allowed to go through the firewall as specified in section [4.3.1] of [RFC 4890]:

- Time Exceeded (Type 3) – Code 0 only
- Parameter Problem (Type 4) – Codes 1 and 2 only
- Echo Request (Type 128)
- Echo Response (Type 129)

Note: Destination Unreachable and Packet Too Big messages MUST be allowed to go through as they are used in some mechanisms to discover path MTU.

It is RECOMMENDED that this service deploys access control as defined in [IGD2] and use it by default. Note that the vendor is free to use a different default or to enable the user to change the default.

It is RECOMMENDED that firewall controlled by this API deploys endpoint independent filtering as specified in [REC-17] and [REC-33] of [RFC6092].

3 Theory of Operation (Informative)

3.1 IPv4 NAT and IPv6 firewall control relationship

In IPv4, NAT (Network Address Translation) is a popular service for alleviating IPv4 address shortage. For example, NAT allows to enable multiple hosts on a private network to access the Internet using a single public IP address.

IPv6 has been designed in part to correct certain deficiencies in IPv4. Especially, the IPv6 address space is much larger than the IPv4 one so there is hopefully no risk of an IPv6 address shortage. As a result, NAT is not needed in IPv6, and an end-to-end communication paradigm is restored.

However, one of the side-effects of using IPv4 NAT is that many vendors and users believe that NAT provides some basic security to devices by hiding the IP address of the device, which is thereby protected against external attacks. Many experts dispute the supposed security claims of NAT. Nevertheless, it is likely that many vendors and users will want a firewall service to shield internal hosts from external access so that internal devices will continue to have a basic security against external attacks.

In the likely scenario where IPv6 firewalls are common, it will be beneficial to some applications if there is a way to dynamically control the firewall in a way that is similar to NAT traversal. In this case, an IPv6 IGD vendor can implement the *WANIPv6FirewallControl* service (instead of the *WANIPConnection* service) to allow UPnP Control Points to control the IGD's firewall.

For UPnP CPs using the former *WANIPConnection* service, the amount of rework needed to use the *WANIPv6FirewallControl* service will hopefully be light as the two services are pretty similar. The following sections give a detailed overview of how to use the services of *WANIPv6FirewallControl*.

3.2 Start-up

The first thing a UPnP CP SHOULD do when it starts, is to invoke *GetFirewallStatus()* action to know if:

- The IPv6 firewall is enabled;
- The UPnP CP is allowed to create IPv6 firewall pinholes.

If the firewall is disabled, the UPnP CP:

- doesn't have to worry about the firewall blocking any inbound connections from remote hosts;
- can start outbound connections to any remote host.

If the firewall is enabled and the UPnP CP is not allowed to create inbound pinholes, the UPnP CP:

- will not be able to receive unsolicited inbound connections from remote hosts;
- can start outbound connections to any remote host.

If the firewall is enabled and the UPnP CP is allowed to create inbound pinholes, a detailed overview of how to proceed is available in section 3.4.

3.3 Outbound pinhole management

3.3.1 Outbound pinhole creation

If a UPnP CP wants to start an outbound connection to a remote host, it can do it without using any UPnP actions as described in Figure 3-1.

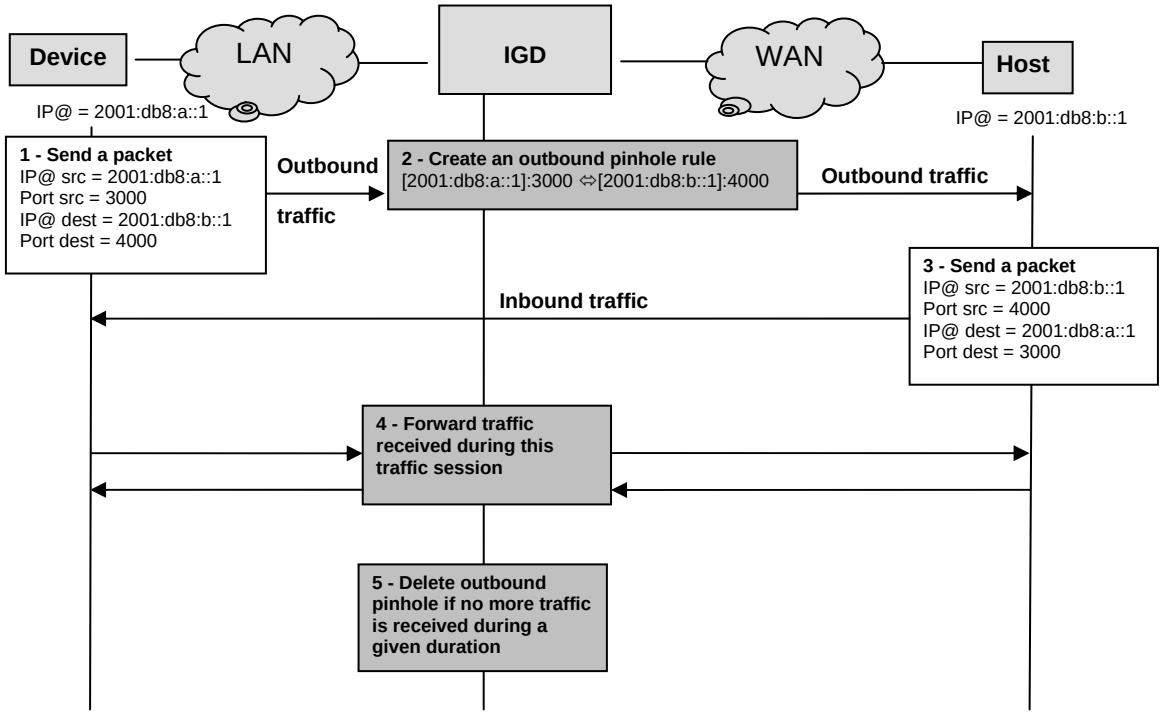

Figure 3-1: Outbound pinhole creation

In Figure 3-1, the IPv6 firewall that is embedded in the functional box labeled “IGD” will dynamically create a pinhole for outbound and inbound traffic associated. This action is not under the control of *WANIPv6FirewallControl* service. Typically inbound traffic from “Host” to “Device” in Figure 3-1 will be allowed for some period of time after the most recent outbound packet is observed by the firewall. It is also typical for the firewall to be “stateful” and understand the elements of procedure of the particular protocol that is used, e.g. TCP.

It is also the case that many firewalls use “Endpoint Independent Filtering” which means that the firewall places no restrictions on the source of address of inbound packets. In the case of Figure 3-1, therefore, inbound packets are not filtered to be from “Host” but can originate from any address when the firewall supports endpoint independent filtering. When the firewall supports endpoint independent filtering, an internal device can open a pinhole for unsolicited packets from any external address by sending a packet to any external address.

3.3.2 Outbound pinhole refresh

In some cases, the UPnP CP wants to keep this outbound pinhole open, so the remote host can send it inbound traffic even when the connection becomes dormant and neither side sends packets for a prolonged period of time. The CP can keep the pinhole open by checking the *LeaseTime* and sending a packet before it expires. Conversely, if this CP wants to optimize energy consumption needed to keep this pinhole open, it can use *GetOutboundPinholeTimeout()* action as described in Figure 3-2.

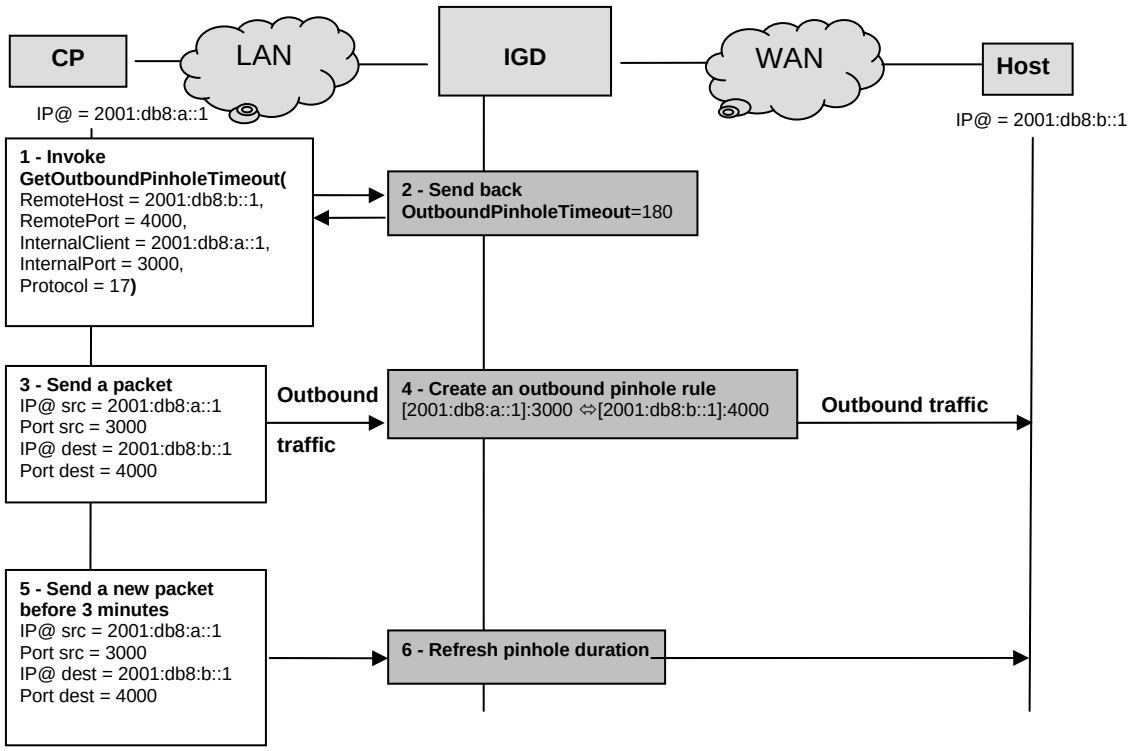

Figure 3-2: Outbound pinhole refresh

3.3.3 Outbound pinhole lifecycle

The Figure 3-3 gives the state transition diagram of an outbound pinhole.

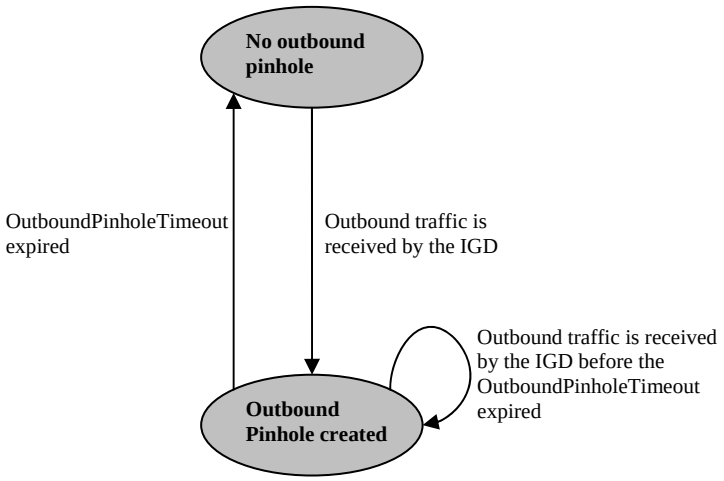

Figure 3-3: Outbound pinhole state transition diagram

3.4 Inbound Pinhole management

3.4.1 Inbound pinhole creation

If a UPnP CP needs to receive unsolicited inbound traffic from a specific remote host address. It can create an inbound pinhole by using *AddPinhole()* action as described in Figure 3-4.

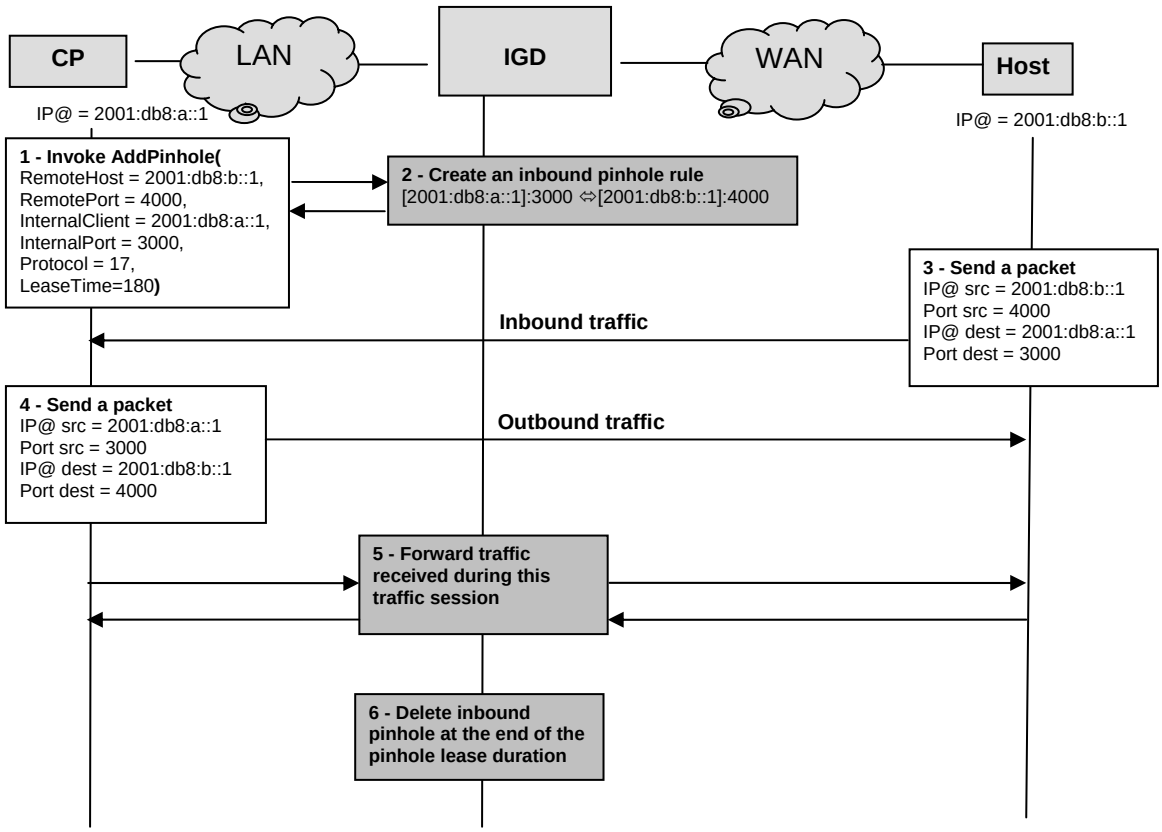

Figure 3-4: Inbound pinhole creation

If the CP wishes to receive unsolicited inbound traffic from any external host, it can use a wildcard value for *RemoteHost* and *RemotePort*. It can similarly wildcard the *RemotePort* if it wishes to receive unsolicited packets from a specific external host IPv6 address from any port.

3.4.2 Checking that an inbound pinhole is working

If the UPnP CP needs to know how many IP packets have been going through its pinhole, it can save the *UniqueID* returned by *AddPinhole()* action and use it in *GetPinholePackets()* action as described in Figure 3-5. If the *PinholePackets* value returned by this action is different than 0, the UPnP CP can know that its pinhole had worked, at least once. Moreover, if the CP retrieves the *PinholePackets* value at a later time, and the value has increased, the CP can suppose that the pinhole is still working.

If the UPnP CP needs to verify that its pinhole allows traffic to pass though the firewall, it can save the *UniqueID* returned by *AddPinhole()* action and use it in *CheckPinholeWorking()* action as described in Figure 3-5.

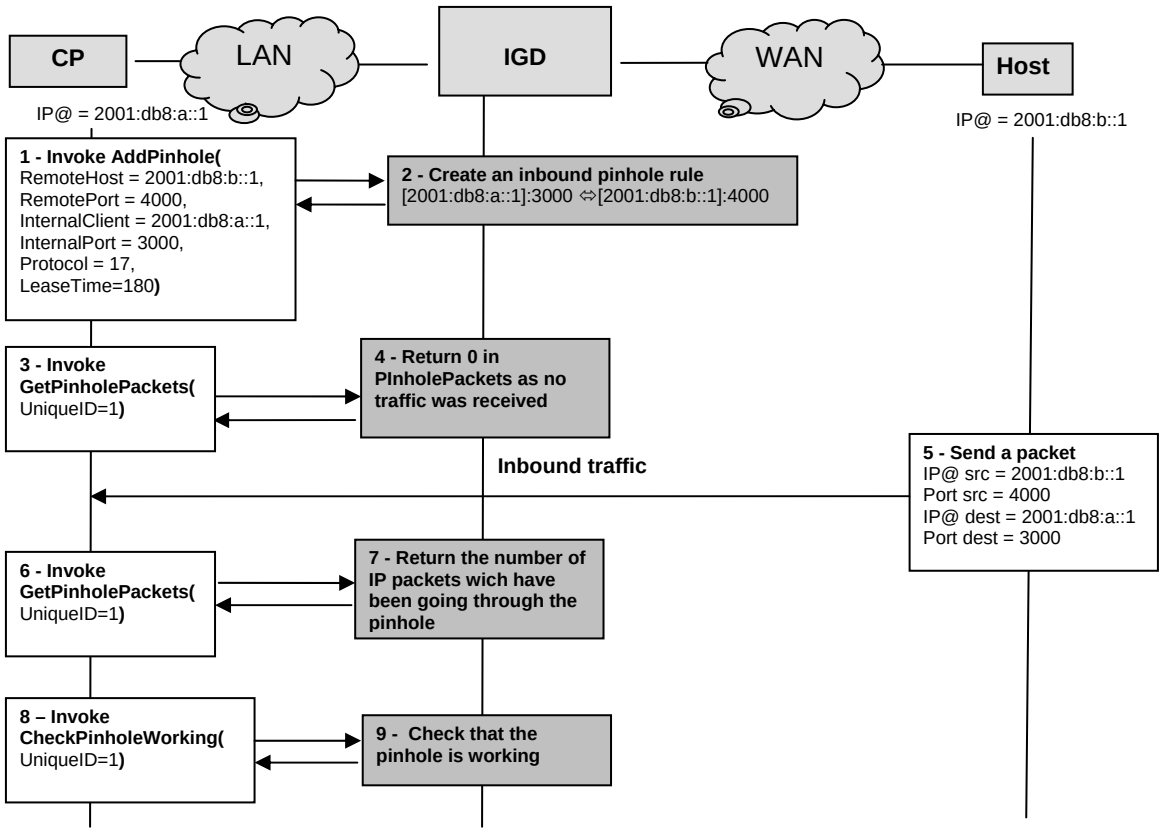

Figure 3-5: Checking that an inbound pinhole is working

3.4.3 Inbound pinhole refresh

If the UPnP CP wants to refresh its pinhole, it can save the *UniqueID* returned by *AddPinhole()* action and use it in *UpdatePinhole()* action as described in Figure 3-6.

If the UPnP CP wants to delete its pinhole before the lease duration expires, it can save the *UniqueID* returned by *AddPinhole()* action and use it in *DeletePinhole()* action as described in Figure 3-6.

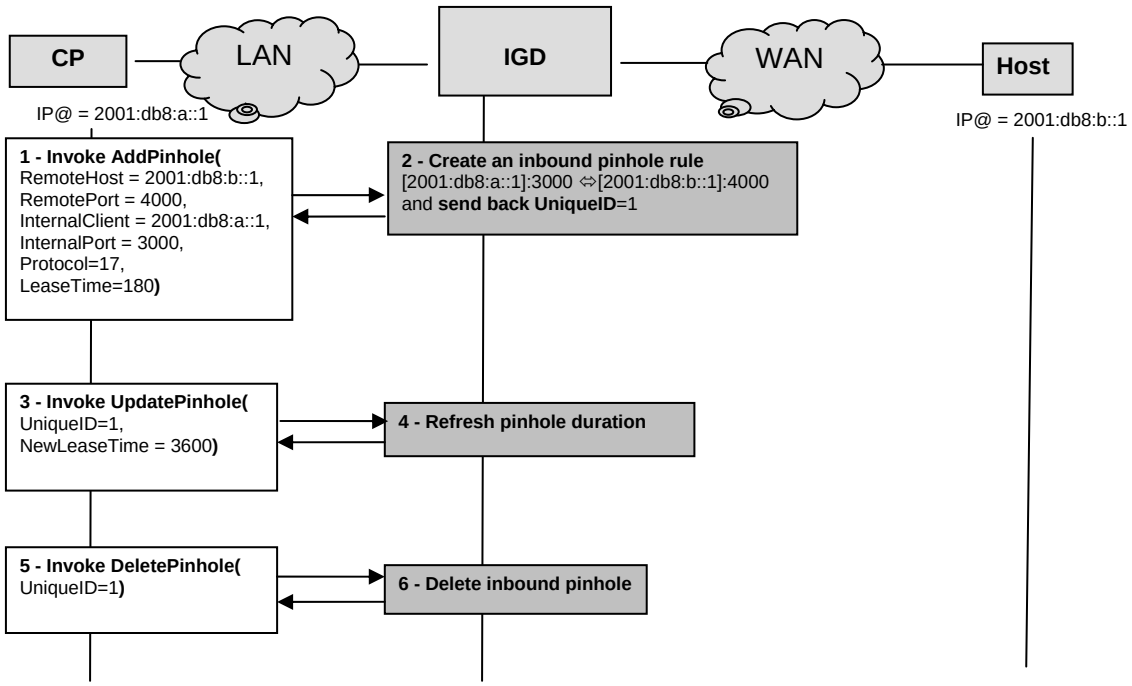

Figure 3-6: Inbound pinhole refresh and deletion

3.4.4 Inbound pinhole state transition diagram

The Figure 3-7 gives the state transition diagram of an inbound pinhole.

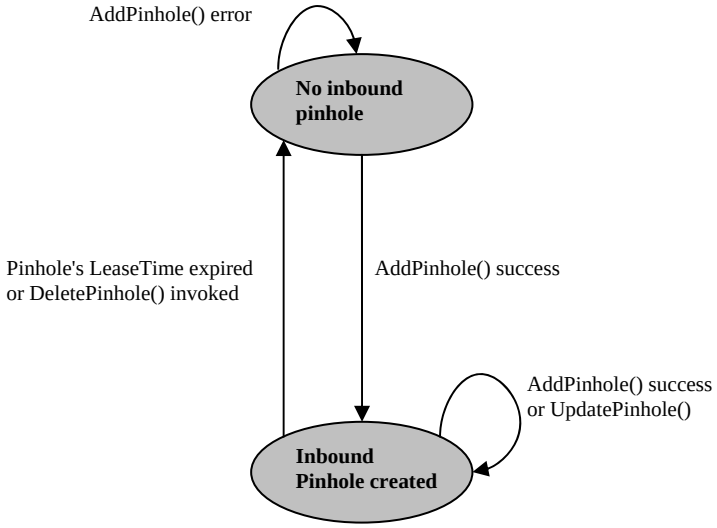

Figure 3-7: Inbound pinhole state transition diagram

4 XML Service Description

```
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">

   <specVersion>
      <major>1</major>
      <minor>0</minor>
   </specVersion>

   <actionList>

      <action>
         <name>GetFirewallStatus</name>
         <argumentList>
            <argument>
               <name>FirewallEnabled</name>
               <direction>out</direction>
               <relatedStateVariable>
                  FirewallEnabled
               </relatedStateVariable>
            </argument>

            <argument>
               <name>InboundPinholeAllowed</name>
               <direction>out</direction>
               <relatedStateVariable>
                  InboundPinholeAllowed
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>

      <action>
         <name>GetOutboundPinholeTimeout</name>
         <argumentList>
            <argument>
               <name>RemoteHost</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_IPv6Address
               </relatedStateVariable>
            </argument>

            <argument>
               <name>RemotePort</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_Port
               </relatedStateVariable>
            </argument>

            <argument>
               <name>InternalClient</name>
               <direction>in</direction>
               <relatedStateVariable>
```

```
                A_ARG_TYPE_IPv6Address
            </relatedStateVariable>
        </argument>

        <argument>
            <name>InternalPort</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_Port
            </relatedStateVariable>
        </argument>

        <argument>
            <name>Protocol</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_Protocol
            </relatedStateVariable>
        </argument>

        <argument>
            <name>OutboundPinholeTimeout</name>
            <direction>out</direction>
            <relatedStateVariable>
                A_ARG_TYPE_OutboundPinholeTimeout
            </relatedStateVariable>
        </argument>
    </argumentList>
</action>

<action>
    <name>AddPinhole</name>
    <argumentList>
        <argument>
            <name>RemoteHost</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_IPv6Address
            </relatedStateVariable>
        </argument>

        <argument>
            <name>RemotePort</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_Port
            </relatedStateVariable>
        </argument>

        <argument>
            <name>InternalClient</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_IPv6Address
            </relatedStateVariable>
        </argument>
```

```
            <argument>
               <name>InternalPort</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_Port
               </relatedStateVariable>
            </argument>

            <argument>
               <name>Protocol</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_Protocol
               </relatedStateVariable>
            </argument>

            <argument>
               <name>LeaseTime</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_LeaseTime
               </relatedStateVariable>
            </argument>

            <argument>
               <name>UniqueID</name>
               <direction>out</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_UniqueID
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>

      <action>
         <name>UpdatePinhole</name>
         <argumentList>
            <argument>
               <name>UniqueID</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_UniqueID
               </relatedStateVariable>
            </argument>

            <argument>
               <name>NewLeaseTime</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_LeaseTime
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>

      <action>
         <name>DeletePinhole</name>
```

```
            <argumentList>
               <argument>
                  <name>UniqueID</name>
                  <direction>in</direction>
                  <relatedStateVariable>
                     A_ARG_TYPE_UniqueID
                  </relatedStateVariable>
               </argument>
            </argumentList>
         </action>

         <action>
            <name>GetPinholePackets</name>
            <argumentList>
               <argument>
                  <name>UniqueID</name>
                  <direction>in</direction>
                  <relatedStateVariable>
                     A_ARG_TYPE_UniqueID
                  </relatedStateVariable>
               </argument>

               <argument>
                  <name>PinholePackets</name>
                  <direction>out</direction>
                  <relatedStateVariable>
                     A_ARG_TYPE_PinholePackets
                  </relatedStateVariable>
               </argument>
            </argumentList>
         </action>


         <action>
            <name>CheckPinholeWorking</name>
            <argumentList>
               <argument>
                  <name>UniqueID</name>
                  <direction>in</direction>
                  <relatedStateVariable>
                     A_ARG_TYPE_UniqueID
                  </relatedStateVariable>
               </argument>

               <argument>
                  <name>IsWorking</name>
                  <direction>out</direction>
                  <relatedStateVariable>
                     A_ARG_TYPE_Boolean
                  </relatedStateVariable>
               </argument>
            </argumentList>
         </action>

      </actionList>

      <serviceStateTable>
```

```
<stateVariable sendEvents="yes">
   <name>FirewallEnabled</name>
   <dataType>boolean</dataType>
</stateVariable>

<stateVariable sendEvents="yes">
   <name>InboundPinholeAllowed</name>
   <dataType>boolean</dataType>
</stateVariable>

<stateVariable sendEvents="no">
   <name>A_ARG_TYPE_OutboundPinholeTimeout</name>
   <dataType>ui4</dataType>
   <allowedValueRange>
      <minimum>Vendor-defined</minimum>
      <maximum>Vendor-defined</maximum>
   </allowedValueRange>
</stateVariable>

<stateVariable sendEvents="no">
   <name>A_ARG_TYPE_IPv6Address</name>
   <dataType>string</dataType>
</stateVariable>

<stateVariable sendEvents="no">
   <name>A_ARG_TYPE_Port</name>
   <dataType>ui2</dataType>
</stateVariable>

<stateVariable sendEvents="no">
   <name>A_ARG_TYPE_Protocol</name>
   <dataType>ui2</dataType>
</stateVariable>

<stateVariable sendEvents="no">
   <name>A_ARG_TYPE_LeaseTime</name>
   <dataType>ui4</dataType>
   <allowedValueRange>
      <minimum>1</minimum>
      <maximum>86400</maximum>
   </allowedValueRange>
</stateVariable>

<stateVariable sendEvents="no">
   <name>A_ARG_TYPE_UniqueID</name>
   <dataType>ui2</dataType>
</stateVariable>

<stateVariable sendEvents="no">
   <name>A_ARG_TYPE_PinholePackets</name>
   <dataType>ui4</dataType>
</stateVariable>

<stateVariable sendEvents="no">
   <name>A_ARG_TYPE_Boolean</name>
   <dataType>boolean</dataType>
```

```
      </stateVariable>

   </serviceStateTable>
</scpd>
```

5 Security Considerations (Informative)

This section is informative and identifies the threats to the *WANIPv6FirewallControl* DCP and the steps taken in the DCP to protect against various threats. This section also considers the policy alternatives for *WANIPv6FirewallControl*, summarizes the recommended policy defaults for this DCP, and discusses the importance of IPv6 firewalls to users and their devices.

Firewall technologies became widespread in the 1980's to protect a local network service from unauthorized access [CB]. By filtering packets, monitoring connection state, and by selectively relaying application packets, firewalls can limit unauthorized access to and from the devices on the home network. A firewall controlled by *WANIPv6FirewallControl* typically offers no protection against attacks that originate from inside the home network but protect network assets from external threats, such as the very common incidents of port scans originating from the Internet. Although some firewalls can filter outbound packets, Section 1 of this document states that the *WANIPv6FirewallControl* service is strictly for control of inbound packets. Thus, *WANIPv6FirewallControl* is a service to control firewalls that protect internal assets from external threats.

5.1 Firewall Assets, Risks and Threats

An Internet Gateway Device has a set of Assets that need to be protected against risks and threats. Most of the attacks originate on the local area network and are thus "internal" to the home network. Attacks that originate from the WAN side of the IGD are generally prevented by a firewall that runs on the IGD device.

Thus, an external attack originates outside the home network. Through an "external attack", some attacker who is outside the home network gains unauthorized access to a home network device. The successful external attacker needs to (1) get access to an internal transport address having (2) an application program running that is listening, receiving and processing messages on that transport address and has (3) no access controls. The combination of an active listener on the inside and an open firewall to the outside can lead to a successful attack on a home network device, such as a server that does not strongly authenticate the access. Although it is generally true that few such active listening applications are running on home network devices today, packet filtering is an effective default setting for internal devices that are not intended for external access. Packet filtering firewalls are found in most commercial router/gateways. But malware can be an active (and malicious!) listener that aids an external attack by using UPnP NAT traversal or Bonjour NAT-PMP to open the firewall for outsiders who are up to no good. This type of attack is found in the recent Conficker virus, for example. Some consider the firewall to be of little value when resident malware can easily open the firewall for outside access. Such internal attack vectors need to be considered when determining certain policies for firewall control such as whether to require authentication or not.

5.2 Firewall Control Policy and Recommendations

UPnP device control protocols provide various mechanisms for controlling network devices. These mechanisms can be configured and used in various ways according to *policy*. One policy decision, for example, is whether to enable the UPnP *WANIPv6FirewallControl* service by default. This section considers these and other defaults, such as the use of authentication of *WANIPv6FirewallControl* actions using the UPnP *DeviceProtection* service. *DeviceProtection* provides a secure introduction method and strong authentication of requests to the IGD for firewall control. The vendor and the user of *WANIPv6FirewallControl* needs to determine whether or not to use *DeviceProtection* to authenticate firewall control requests to establish a pinhole.

Thus, the vendor of an IGD has to make several major policy decisions regarding an IPv6 firewall.

1. Whether or not an IPv6 firewall will be shipped with *WANIP6FirewallControl*:

   This document makes no recommendation regarding use or non-use of an IPv6 firewall in an Internet gateway. For purely technical reasons, it is infeasible for an outsider to do an effective port scan on IPv6 global unicast addresses since there are 2^{64} possible addresses and it will take centuries to reliably guess a valid address on the home network using today's computer and packet-network technologies.

Thus, the most common reason for filtering unsolicited packets from outside the firewalled network is eliminated: Port scans are infeasible even when there is an application-layer process listening on a port. Only nodes that are publicly advertised on the IPv6 Internet are accessible from the IPv6 Internet. If a home-network address is advertised in the DNS or if the traffic from the home-network is monitored from outside the home network such as through P2P trackers or HTTP services, then this address could be easily accessed from the IPv6 Internet[1]. Thus, if the gateway is run without an IPv6 firewall, then hosts with public addresses will need to control network access through such means as usernames and passwords. Of course if there are no applications listening on the public IPv6 transport address, then such port scans will not succeed. In case there are, it still might be possible to forgo the use of a home-network firewall when all IPv6 addresses used outside the home network are temporary. This topic is outside the scope of this DCP, and this document makes no recommendation.

2. Whether or not the firewall is enabled by default:

For the reasons stated under item #1, above, we make no recommendation regarding whether a firewall that is shipped in a gateway product is enabled or not.

3. Whether or not *WANIPv6FirewallControl* is enabled by default:

This document makes no recommendations regarding use of *WANIPv6FirewallControl* – whether it is shipped in the device or enabled by default.

4. Whether or not *WANIPv6FirewallControl* actions such as pinhole creation are authenticated by default:

As stated above in this section, UPnP *DeviceProtection* offers limited protection against malware, which on some control points might be able to access the credentials needed to authenticate and authorize the opening of the firewall. Still, *DeviceProtection* makes an attack on firewall control more difficult, and it is possible to use both public key cryptography and a password to authenticate a firewall control session. For this reason section 2 of this document recommends use of *DeviceProtection* using public key cryptography and optionally password access control (see the service requirements for the various pinhole actions).

5. Whether or not the defaults can be changed by an authorized user:

This document makes no recommendation regarding whether or not the vendor chooses to allow the authorized user to change any of the defaults.

To summarize, this document only makes one policy recommendation: If the vendor ships an IPv6 firewall with *WANIPv6FirewallControl*, we recommend that UPnP *DeviceProtection* be used *by default* for the creation of firewall pinholes. The default configuration of UPnP *DeviceProtection* for this service is explained in section 2 (see the service requirements for the various pinhole actions).

[1] Temporary IPv6 addresses can shrink the window of time during which a harvested address can be used b an attacker.