INTERNATIONAL STANDARD

ISO/IEC 29341-24-11

First edition
2017-09

Information technology — UPnP Device Architecture —

Part 24-11: Internet gateway device control protocol — Level 2 — Wide area network internet protocol v6 — Firewall control service

Technologies de l'information — Architecture de dispositif UPnP — Partie 24-11: Protocole de contrôle de dispositif de passerelle Internet — Niveau 2 — Protocole internet de réseau étendu v6 — Service de contrôle du pare-feu

Reference number
ISO/IEC 29341-24-11:2017(E)

COPYRIGHT PROTECTED DOCUMENT

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

CONTENTS

1 Scope 1
2 Normative References 1
3 Terms, definitions, symbols and abbreviations 2
4 Notations and conventions 4
4.1 Notation 4
4.2 Data types 4
4.3 Vendor-defined extensions 4
5 Service Model 4
5.1 Service Type 4
5.2 Service Architecture 5
5.3 State Variables 5
5.3.1 Summary 5
5.3.2 *FirewallEnabled* 5
5.3.3 *InboundPinholeAllowed* 6
5.3.4 *A_ARG_TYPE_OutboundPinholeTimeout* 6
5.3.5 *A_ARG_TYPE_IPv6Address* 6
5.3.6 *A_ARG_TYPE_Port* 6
5.3.7 *A_ARG_TYPE_Protocol* 6
5.3.8 *A_ARG_TYPE_LeaseTime* 6
5.3.9 *A_ARG_TYPE_UniqueID* 7
5.3.10 *A_ARG_TYPE_PinholePackets* 7
5.3.11 *A_ARG_TYPE_Boolean* 7
5.3.12 Relationships among State Variables 7
5.4 Eventing and Moderation 7
5.4.1 Summary 7
5.4.2 Eventing of *FirewallEnabled* 7
5.4.3 Eventing of *InboundPinholeAllowed* 7
5.5 Actions 7
5.5.1 Summary 7
5.5.2 *GetFirewallStatus()* 8
5.5.3 *GetOutboundPinholeTimeout()* 8
5.5.4 *AddPinhole()* 10
5.5.5 *UpdatePinhole()* 12
5.5.6 *DeletePinhole()* 13
5.5.7 *GetPinholePackets()* 14
5.5.8 *CheckPinholeWorking()* 15
5.5.9 Relationships Between Actions 17
5.5.10 Error Code Summary 17
5.6 Service Behavioral Model 17
6 XML Service Description 18
Annex A (informative) Theory of Operation 23
A.1 IPv4 NAT and IPv6 firewall control relationship 23
A.2 Start-up 23
A.3 Outbound pinhole management 24
A.3.1 Outbound pinhole creation 24
A.3.2 Outbound pinhole refresh 24

A.3.3 Outbound pinhole lifecycle 25
A.4 Inbound Pinhole management 25
A.4.1 Inbound pinhole creation 25
A.4.2 Checking that an inbound pinhole is working 26
A.4.3 Inbound pinhole refresh 27
A.4.4 Inbound pinhole state transition diagram 28
Annex B (normative) Security Considerations 29
B.1 Overview 29
B.2 Firewall Assets, Risks and Threats 29
B.3 Firewall Control Policy and Recommendations 29
Annex C (informative) Bibliography 31

Figure A.1 — Outbound pinhole creation 24
Figure A.2 — Outbound pinhole refresh 25
Figure A.3 — Outbound pinhole state transition diagram 25
Figure A.4 — Inbound pinhole creation 26
Figure A.5 — Checking that an inbound pinhole is working 27
Figure A.6 — Inbound pinhole refresh and deletion 28
Figure A.7 — Inbound pinhole state transition diagram 28

Table 1 — State Variables 5
Table 2 — allowedValueRange for *A_ARG_TYPE_OutboundPinholeTimeout* 6
Table 3 — allowedValueRange for *A_ARG_TYPE_LeaseTime* 6
Table 4 — Eventing and Moderation 7
Table 5 — Actions 7
Table 6 — Arguments for *GetFirewallStatus()* 8
Table 7 — Error Codes for *GetFirewallStatus()* 8
Table 8 — Arguments for *GetOutboundPinholeTimeout()* 9
Table 9 — Error Codes for *GetOutboundPinholeTimeout()* 10
Table 10 — Arguments for *AddPinhole()* 10
Table 11 — Error Codes for *AddPinhole()* 11
Table 12 — Arguments for *UpdatePinhole()* 12
Table 13 — Error Codes for *UpdatePinhole()* 13
Table 14 — Arguments for *DeletePinhole()* 13
Table 15 — Error Codes for *DeletePinhole()* 14
Table 16 — Arguments for *GetPinholePackets()* 14
Table 17 — Error Codes for *GetPinholePackets()* 15
Table 18 — Arguments for *CheckPinholeWorking()* 16
Table 19 — Error Codes for *CheckPinholeWorking()* 16
Table 20 — Error Code Summary 17

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see http://www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the voluntary nature of Standard, the meaning of the ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword – Supplementary information

ISO/IEC 29341-24-11was prepared by UPnP Forum and adopted, under the PAS procedure, by joint technical committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC.

The list of all currently available parts of ISO/IEC 29341 series, under the general title *Information technology — UPnP Device Architecture*, can be found on the ISO web site.

Introduction

ISO and IEC draw attention to the fact that it is claimed that compliance with this document may involve the use of patents as indicated below.

ISO and IEC take no position concerning the evidence, validity and scope of these patent rights. The holders of -these patent rights have assured ISO and IEC that they are willing to negotiate licenses under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of these patent rights are registered with ISO and IEC.

Intel Corporation has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Intel Corporation
Standards Licensing Department
5200 NE Elam Young Parkway
MS: JFS-98
USA – Hillsboro, Oregon 97124

Microsoft Corporation has informed IEC and ISO that it has patent applications or granted patents as listed below:

6101499 / US; 6687755 / US; 6910068 / US; 7130895 / US; 6725281 / US; 7089307 / US; 7069312 / US; 10/783 524 /US

Information may be obtained from:

Microsoft Corporation
One Microsoft Way
USA – Redmond WA 98052

Philips International B.V. has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Philips International B.V. – IP&S
High Tech campus, building 44 3A21
NL – 5656 Eindhoven

NXP B.V. (NL) has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

NXP B.V. (NL)
High Tech campus 60
NL – 5656 AG Eindhoven

Matsushita Electric Industrial Co. Ltd. has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Matsushita Electric Industrial Co. Ltd.
1-3-7 Shiromi, Chuoh-ku
JP – Osaka 540-6139

Hewlett Packard Company has informed IEC and ISO that it has patent applications or granted patents as listed below:

5 956 487 / US; 6 170 007 / US; 6 139 177 / US; 6 529 936 / US; 6 470 339 / US; 6 571 388 / US; 6 205 466 / US

Information may be obtained from:

Hewlett Packard Company
1501 Page Mill Road
USA – Palo Alto, CA 94304

Samsung Electronics Co. Ltd. has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Digital Media Business, Samsung Electronics Co. Ltd.
416 Maetan-3 Dong, Yeongtang-Gu,
KR – Suwon City 443-742

Huawei Technologies Co., Ltd. has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Huawei Technologies Co., Ltd.
Administration Building, Bantian Longgang District
Shenzhen – China 518129

Qualcomm Incorporated has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Qualcomm Incorporated
5775 Morehouse Drive
San Diego, CA – USA 92121

Telecom Italia S.p.A.has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Telecom Italia S.p.A.
Via Reiss Romoli, 274
Turin - Italy 10148

Cisco Systems informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA – USA 95134

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Original UPnP Document

Reference may be made in this document to original UPnP documents. These references are retained in order to maintain consistency between the specifications as published by ISO/IEC and by UPnP Implementers Corporation and later by UPnP Forum. The following table indicates the original UPnP document titles and the corresponding part of ISO/IEC 29341:

UPnP Document Title	ISO/IEC 29341 Part
UPnP Device Architecture 1.0	ISO/IEC 29341-1:2008
UPnP Device Architecture Version 1.0	ISO/IEC 29341-1:2011
UPnP Device Architecture 1.1	ISO/IEC 29341-1-1:2011
UPnP Device Architecture 2.0	ISO/IEC 29341-1-2
UPnP Basic:1 Device	ISO/IEC 29341-2
UPnP AV Architecture:1	ISO/IEC 29341-3-1:2008
UPnP AV Architecture:1	ISO/IEC 29341-3-1:2011
UPnP AVTransport:1 Service	ISO/IEC 29341-3-10
UPnP ConnectionManager:1 Service	ISO/IEC 29341-3-11
UPnP ContentDirectory:1 Service	ISO/IEC 29341-3-12
UPnP RenderingControl:1 Service	ISO/IEC 29341-3-13
UPnP MediaRenderer:1 Device	ISO/IEC 29341-3-2
UPnP MediaRenderer:2 Device	ISO/IEC 29341-3-2:2011
UPnP MediaServer:1 Device	ISO/IEC 29341-3-3
UPnP AVTransport:2 Service	ISO/IEC 29341-4-10:2008
UPnP AVTransport:2 Service	ISO/IEC 29341-4-10:2011
UPnP ConnectionManager:2 Service	ISO/IEC 29341-4-11:2008
UPnP ConnectionManager:2 Service	ISO/IEC 29341-4-11:2011
UPnP ContentDirectory:2 Service	ISO/IEC 29341-4-12
UPnP RenderingControl:2 Service	ISO/IEC 29341-4-13:2008
UPnP RenderingControl:2 Service	ISO/IEC 29341-4-13:2011
UPnP ScheduledRecording:1	ISO/IEC 29341-4-14
UPnP ScheduledRecording:2	ISO/IEC 29341-4-14:2011

UPnP MediaRenderer:2 Device	ISO/IEC 29341-4-2
UPnP MediaServer:2 Device	ISO/IEC 29341-4-3
UPnP AV Datastructure Template:1	ISO/IEC 29341-4-4:2008
UPnP AV Datastructure Template:1	ISO/IEC 29341-4-4:2011
UPnP DigitalSecurityCamera:1 Device	ISO/IEC 29341-5-1
UPnP DigitalSecurityCameraMotionImage:1 Service	ISO/IEC 29341-5-10
UPnP DigitalSecurityCameraSettings:1 Service	ISO/IEC 29341-5-11
UPnP DigitalSecurityCameraStillImage:1 Service	ISO/IEC 29341-5-12
UPnP HVAC_System:1 Device	ISO/IEC 29341-6-1
UPnP ControlValve:1 Service	ISO/IEC 29341-6-10
UPnP HVAC_FanOperatingMode:1 Service	ISO/IEC 29341-6-11
UPnP FanSpeed:1 Service	ISO/IEC 29341-6-12
UPnP HouseStatus:1 Service	ISO/IEC 29341-6-13
UPnP HVAC_SetpointSchedule:1 Service	ISO/IEC 29341-6-14
UPnP TemperatureSensor:1 Service	ISO/IEC 29341-6-15
UPnP TemperatureSetpoint:1 Service	ISO/IEC 29341-6-16
UPnP HVAC_UserOperatingMode:1 Service	ISO/IEC 29341-6-17
UPnP HVAC_ZoneThermostat:1 Device	ISO/IEC 29341-6-2
UPnP BinaryLight:1 Device	ISO/IEC 29341-7-1
UPnP Dimming:1 Service	ISO/IEC 29341-7-10
UPnP SwitchPower:1 Service	ISO/IEC 29341-7-11
UPnP DimmableLight:1 Device	ISO/IEC 29341-7-2
UPnP InternetGatewayDevice:1 Device	ISO/IEC 29341-8-1
UPnP LANHostConfigManagement:1 Service	ISO/IEC 29341-8-10
UPnP Layer3Forwarding:1 Service	ISO/IEC 29341-8-11
UPnP LinkAuthentication:1 Service	ISO/IEC 29341-8-12
UPnP RadiusClient:1 Service	ISO/IEC 29341-8-13

UPnP WANCableLinkConfig:1 Service	ISO/IEC 29341-8-14
UPnP WANCommonInterfaceConfig:1 Service	ISO/IEC 29341-8-15
UPnP WANDSLLinkConfig:1 Service	ISO/IEC 29341-8-16
UPnP WANEthernetLinkConfig:1 Service	ISO/IEC 29341-8-17
UPnP WANIPConnection:1 Service	ISO/IEC 29341-8-18
UPnP WANPOTSLinkConfig:1 Service	ISO/IEC 29341-8-19
UPnP LANDevice:1 Device	ISO/IEC 29341-8-2
UPnP WANPPPConnection:1 Service	ISO/IEC 29341-8-20
UPnP WLANConfiguration:1 Service	ISO/IEC 29341-8-21
UPnP WANDevice:1 Device	ISO/IEC 29341-8-3
UPnP WANConnectionDevice:1 Device	ISO/IEC 29341-8-4
UPnP WLANAccessPointDevice:1 Device	ISO/IEC 29341-8-5
UPnP Printer:1 Device	ISO/IEC 29341-9-1
UPnP ExternalActivity:1 Service	ISO/IEC 29341-9-10
UPnP Feeder:1.0 Service	ISO/IEC 29341-9-11
UPnP PrintBasic:1 Service	ISO/IEC 29341-9-12
UPnP Scan:1 Service	ISO/IEC 29341-9-13
UPnP Scanner:1.0 Device	ISO/IEC 29341-9-2
UPnP QoS Architecture:1.0	ISO/IEC 29341-10-1
UPnP QosDevice:1 Service	ISO/IEC 29341-10-10
UPnP QosManager:1 Service	ISO/IEC 29341-10-11
UPnP QosPolicyHolder:1 Service	ISO/IEC 29341-10-12
UPnP QoS Architecture:2	ISO/IEC 29341-11-1
UPnP QosDevice:2 Service	ISO/IEC 29341-11-10
UPnP QosManager:2 Service	ISO/IEC 29341-11-11
UPnP QosPolicyHolder:2 Service	ISO/IEC 29341-11-12
UPnP QOS v2 Schema Files	ISO/IEC 29341-11-2

UPnP RemoteUIClientDevice:1 Device	ISO/IEC 29341-12-1
UPnP RemoteUIClient:1 Service	ISO/IEC 29341-12-10
UPnP RemoteUIServer:1 Service	ISO/IEC 29341-12-11
UPnP RemoteUIServerDevice:1 Device	ISO/IEC 29341-12-2
UPnP DeviceSecurity:1 Service	ISO/IEC 29341-13-10
UPnP SecurityConsole:1 Service	ISO/IEC 29341-13-11
UPnP ContentDirectory:3 Service	ISO/IEC 29341-14-12:2011
UPnP MediaServer:3 Device	ISO/IEC 29341-14-3:2011
UPnP ContentSync:1	ISO/IEC 29341-15-10:2011
UPnP Low Power Architecture:1	ISO/IEC 29341-16-1:2011
UPnP LowPowerProxy:1 Service	ISO/IEC 29341-16-10:2011
UPnP LowPowerDevice:1 Service	ISO/IEC 29341-16-11:2011
UPnP QoS Architecture:3	ISO/IEC 29341-17-1:2011
UPnP QosDevice:3 Service	ISO/IEC 29341-17-10:2011
UPnP QosManager:3 Service	ISO/IEC 29341-17-11:2011
UPnP QosPolicyHolder:3 Service	ISO/IEC 29341-17-12:2011
UPnP QosDevice:3 Addendum	ISO/IEC 29341-17-13:2011
UPnP RemoteAccessArchitecture:1	ISO/IEC 29341-18-1:2011
UPnP InboundConnectionConfig:1 Service	ISO/IEC 29341-18-10:2011
UPnP RADAConfig:1 Service	ISO/IEC 29341-18-11:2011
UPnP RADASync:1 Service	ISO/IEC 29341-18-12:2011
UPnP RATAConfig:1 Service	ISO/IEC 29341-18-13:2011
UPnP RAClient:1 Device	ISO/IEC 29341-18-2:2011
UPnP RAServer:1 Device	ISO/IEC 29341-18-3:2011
UPnP RADiscoveryAgent:1 Device	ISO/IEC 29341-18-4:2011
UPnP SolarProtectionBlind:1 Device	ISO/IEC 29341-19-1:2011
UPnP TwoWayMotionMotor:1 Service	ISO/IEC 29341-19-10:2011

UPnP AV Architecture:2	ISO/IEC 29341-20-1
UPnP AVTransport:3 Service	ISO/IEC 29341-20-10
UPnP ConnectionManager:3 Service	ISO/IEC 29341-20-11
UPnP ContentDirectory:4 Device	ISO/IEC 29341-20-12
UPnP RenderingControl:3 Service	ISO/IEC 29341-20-13
UPnP ScheduledRecording:2 Service	ISO/IEC 29341-20-14
UPnP MediaRenderer:3 Service	ISO/IEC 29341-20-2
UPnP MediaServer:4 Device	ISO/IEC 29341-20-3
UPnP AV Datastructure Template:1	ISO/IEC 29341-20-4
UPnP InternetGatewayDevice:2 Device	ISO/IEC 29341-24-1
UPnP WANIPConnection:2 Service	ISO/IEC 29341-24-10
UPnP WANIPv6FirewallControl:1 Service	ISO/IEC 29341-24-11
UPnP WANConnectionDevice:2 Service	ISO/IEC 29341-24-2
UPnP WANDevice:2 Device	ISO/IEC 29341-24-3
UPnP Telephony Architecture:2	ISO/IEC 29341-26-1
UPnP CallManagement:2 Service	ISO/IEC 29341-26-10
UPnP MediaManagement:2 Service	ISO/IEC 29341-26-11
UPnP Messaging:2 Service	ISO/IEC 29341-26-12
UPnP PhoneManagement:2 Service	ISO/IEC 29341-26-13
UPnP AddressBook:1 Service	ISO/IEC 29341-26-14
UPnP Calendar:1 Service	ISO/IEC 29341-26-15
UPnP Presense:1 Service	ISO/IEC 29341-26-16
UPnP TelephonyClient:2 Device	ISO/IEC 29341-26-2
UPnP TelephonyServer:2 Device	ISO/IEC 29341-26-3
UPnP Friendly Info Update:1 Service	ISO/IEC 29341-27-1
UPnP MultiScreen MultiScreen Architecture:1	ISO/IEC 29341-28-1
UPnP MultiScreen Application Management:1 Service	ISO/IEC 29341-28-10

UPnP MultiScreen Screen:1 Device	ISO/IEC 29341-28-2
UPnP MultiScreen Application Management:2 Service	ISO/IEC 29341-29-10
UPnP MultiScreen Screen:2 Device	ISO/IEC 29341-29-2
UPnP IoT Management and Control Architecture Overview:1	ISO/IEC 29341-30-1
UPnP DataStore:1 Service	ISO/IEC 29341-30-10
UPnP IoT Management and Control Data Model:1 Service	ISO/IEC 29341-30-11
UPnP IoT Management and Control Transport Generic:1 Service	ISO/IEC 29341-30-12
UPnP IoT Management and Control:1 Device	ISO/IEC 29341-30-2
UPnP Energy Management:1 Service	ISO/IEC 29341-31-1

1 Scope

This document specifies the characteristics of the UPnP networked service named *WANIPv6FirewallControl*, version *1*. This service definition is compliant with *UPnP Service Architecture 1.0* [1].

This service supports the management IPv6 firewall pinholes in Internet gateway devices independent of the WAN access type (PPP, DHCP, ...). It also supports access control to restrict operations to authorized control points and users. It enables UPnP control points to:

- create and delete pinholes of limited duration that allow external communication from the WAN side of the gateway to reach devices on the LAN side
- determine operational status of a pinhole

This service does not support:

- retrieving detailed information about pinholes (such as associated external port or remaining lease duration)
- outbound packet filtering
- modification of security policies of a gateway device

2 Normative References

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

[1] *UPnP Device Architecture, Version 1.0*, UPnP Forum.
Available at: http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0.pdf.

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

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

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

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

[6] 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.

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

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

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

[10] 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.

[11] 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.

[12] *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.

[13] *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.

3 Terms, definitions, symbols and abbreviations

For the purposes of this document, the terms and definitions given in [1] and the following apply:

3.1
firewall
a hardware or software system that inspects network traffic passing through it, and either denies or permits the passage based on a set of rules. It has an "internal" interface (to a LAN) and an "external" interface (normally to a WAN, but sometimes to a different LAN).

3.2
pinhole
a rule that defines an opening in 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.

3.3
inbound traffic
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.

3.4
outbound traffic
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.

3.5
inbound pinhole
a rule created in the firewall to allow inbound initiated traffic to pass through the firewall.

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

automatic pinhole
outbound pinhole that is created automatically when outbound traffic is initiated

3.6
traffic session
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.

3.7 Symbols

3.7.1
::
signifies a hierarchical relationship (parent::child) between the two objects separated by the double colon. This delimiter is used in multiple contexts, such as: Service::Action(), Action()::Argument, parentProperty::childProperty.

3.8 Abbreviations

3.8.1
DHCP
Dynamic Host Configuration Protocol

3.8.2
DNS
Domain Name Service

3.8.3
GUA
Global Unicast Address

3.8.4
HTTP
HyperText Transfer Protocol

3.8.5
IANA
Internet Assigned Numbers Authority

3.8.6
ICMPv6
Internet Control Message Protocol version 6

3.8.7
IGD
Internet Gateway Device

3.8.8
IPv6
Internet Protocol version 6

3.8.9
LAN
Local Area Network

3.8.10
MTU
Maximum Transmission Unit

3.8.11
NAT
Network Address Translation

3.8.12
P2P
Peer to Peer

3.8.13
PMP
Port Mapping Protocol

3.8.14
PPP
Point to Point Protocol

3.8.15
TCP
Transmission Control Protocol

3.8.16
UDP
User Datagram Protocol

3.8.17
WAN
Wide Area Network

4 Notations and conventions

4.1 Notation

- UPnP interface names defined in [1] are styled in **green bold underlined** text.
- UPnP interface names defined outside of [1] are styled in *red italic underlined* text.
- Some additional non-interface names and terms are styled in *italic* text.
- Words that are emphasized are also styled in *italic* text. The difference between italic terms and italics for emphasis will be apparent by context.
- Strings that are to be taken literally are enclosed in “double quotes”.

4.2 Data types

Data type definitions come from two sources:

- All state variable and action argument data types are defined in [1].
- Basic data types for XML element and attribute values are defined in [13]. The XML data types are essential, since all communications between UPnP services and control points are encoded in XML.

For the UPnP-defined data type **boolean** (see [1]), it is strongly recommended to use the value “**0**” for false, and the value “**1**” for true. The values “**true**”, “**false**”, “**yes**” and “**no**” are deprecated—they shall not appear in output, but shall be accepted on input.

For the data type `boolean` defined in [13], it is strongly recommended to use the value “`0`” for false, and the value “`1`” for true. The values “`true`” and “`false`” are allowed, but not recommended.

4.3 Vendor-defined extensions

When vendors add vendor-defined state variables, actions or properties, their assigned names and XML representation shall follow the naming conventions and XML rules for non-standard vendor extensions in [1].

5 Service Model

5.1 Service Type

A service that complies with this specification shall identify itself with the URN:

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

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

5.2 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:

a) 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. Control points identify pinholes by the *UniqueID* value that they retrieve when creating a pinhole.

b) 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.

c) *DeletePinhole()*—removes pinholes.

d) *GetOutboundPinholeTimeout()* returns the default duration value for automatic pinholes.

e) *GetFirewallStatus()* lets control points know if the firewall is enabled and if pinholes can be created through UPnP.

f) *GetPinholePackets()* lets control points get the total number of IP packets which have been going through a certain pinhole.

g) *CheckPinholeWorking()* is an allowed action that enables checking if a certain pinhole allows traffic to pass through the firewall.

5.3 State Variables

5.3.1 Summary

There are two state variables that describe the status of the firewall—whether the firewall is enabled and whether pinholes may be created. All others are virtual state variables, which define the characteristics of various action arguments.

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

Table 1 — State Variables

Variable Name	R/A	Data Type	Reference
FirewallEnabled	*R*	**boolean**	See 5.3.2
InboundPinholeAllowed	*R*	**boolean**	See 5.3.3
A_ARG_TYPE_OutboundPinholeTimeout	*A*	**ui4**	See 5.3.4
A_ARG_TYPE_IPv6Address	*R*	**string**	See 5.3.5
A_ARG_TYPE_Port	*R*	**ui2**	See 5.3.6
A_ARG_TYPE_Protocol	*R*	**ui2**	See 5.3.7
A_ARG_TYPE_LeaseTime	*R*	**ui4**	See 5.3.8
A_ARG_TYPE_UniqueID	*R*	**ui2**	See 5.3.9
A_ARG_TYPE_PinholePackets	*R*	**ui4**	See 5.3.10
A_ARG_TYPE_Boolean	*A*	**boolean**	See 5.3.11
Non standard state variables implemented by a UPnP vendor go here	*X*	*TBD*	*TBD*

5.3.2 *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 control points 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 control points should check *InboundPinholeAllowed* state variable to know if the IGD allows UPnP control points to create inbound pinholes in the firewall.

5.3.3 *InboundPinholeAllowed*

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

If this variable is set to "1" (true), the IGD allows UPnP control points 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 control points 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.

5.3.4 *A_ARG_TYPE_OutboundPinholeTimeout*

Table 2 — allowedValueRange for *A_ARG_TYPE_OutboundPinholeTimeout*

Subelement	Value	R/A
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 [6], 3.2.2).

5.3.5 *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 [8], as domain name defined in [10] or as a wildcard (an empty string).

5.3.6 *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.

5.3.7 *A_ARG_TYPE_Protocol*

This variable is of type **ui2** and its value enumerates the protocol of the pinhole. The specific values shall 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.

5.3.8 *A_ARG_TYPE_LeaseTime*

Table 3 — allowedValueRange for *A_ARG_TYPE_LeaseTime*

Subelement	Value	R/A
minimum	*1*	*R*
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.

5.3.9 *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.

5.3.10 *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.

5.3.11 *A_ARG_TYPE_Boolean*

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

5.3.12 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.

5.4 Eventing and Moderation

5.4.1 Summary

Table 4 — Eventing and Moderation

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

5.4.2 Eventing of *FirewallEnabled*

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

5.4.3 Eventing of *InboundPinholeAllowed*

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

5.5 Actions

5.5.1 Summary

Table 5 — Actions

Name	Service Support R/A	Control Point Support R/A
GetFirewallStatus()	*R*	*A*
GetOutboundPinholeTimeout()	*A*	*A*
AddPinhole()	*R*	*R*
UpdatePinhole()	*R*	*R*
DeletePinhole()	*R*	*A*
GetPinholePackets()	*R*	*A*

Name	Service Support R/A	Control Point Support R/A
CheckPinholeWorking()	*A*	*A*
Non-standard actions implemented by an UPnP vendor go here.	*X*	*X*

5.5.2 *GetFirewallStatus()*

5.5.2.1 Description

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

5.5.2.2 Arguments

Table 6 — Arguments for *GetFirewallStatus()*

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

5.5.2.3 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.

5.5.2.4 Service Requirements

Before processing the action request, the service shall 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 service shall return the 606 *Action not authorized* error code (defined in [1]). [2] recommends access control requirements and authentication levels to be applied by *default* for this action. However, services may choose a different security policy.

5.5.2.5 Control Point Requirements When Calling The Action

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

5.5.2.6 Dependency on Service State

None.

5.5.2.7 Effect on Service State

None.

5.5.2.8 Errors

Table 7 — Error Codes for *GetFirewallStatus()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
600-699	TBD	See Control clause in [1].
606	*Action not authorized*	The action requested requires authorization and the sender was not authorized.

5.5.3 *GetOutboundPinholeTimeout()*

5.5.3.1 Description

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 [6]. 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.

5.5.3.2 Arguments

Table 8 — 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*

5.5.3.3 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".

5.5.3.4 Service Requirements

Before processing the action request, the service shall 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 service shall return the 606 *Action not authorized* error code (defined in [1]). [2] recommends access control requirements and authentication levels to be applied by *default* for this action. However, services 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 allowed. If the requested protocol is not supported, service shall return error 705 *ProtocolNotSupported*.

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

5.5.3.5 Control Point Requirements When Calling The Action

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

5.5.3.6 Dependency on Service State

FirewallEnabled shall be true indicating that firewall is active.

5.5.3.7 Effect on Service State

None.

5.5.3.8 Errors

Table 9 — Error Codes for *GetOutboundPinholeTimeout()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
600-699	TBD	See Control clause in [1].
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.

5.5.4 *AddPinhole()*

5.5.4.1 Description

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.

5.5.4.2 Arguments

Table 10 — Arguments for *AddPinhole()*

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*
LeaseTime	*IN*	*A_ARG_TYPE_LeaseTime*
UniqueID	*OUT*	*A_ARG_TYPE_UniqueID*

5.5.4.3 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 service 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 service 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.

5.5.4.4 Service Requirements

Before processing the action request, the service shall 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 service shall return the 606 *Action not authorized* error code (defined in [1]).

[2] recommends access control requirements and authentication levels to be applied by *default* for this action. However, services may choose a different security policy.

In particular, [2] 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 service shall extend the existing pinhole's lease time and return the *UniqueID* of the existing pinhole.

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.

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.

5.5.4.5 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 [2] should use their IPv6 GUA when calling this action.

5.5.4.6 Dependency on Service State

FirewallEnabled shall be true indicating that firewall is active.

InboundPinholeAllowed shall be true indicating that UPnP control points can create inbound pinhole.

Also, the control point needs to have appropriate access rights to complete this action.

5.5.4.7 Effect on Service State

A new firewall pinhole is created.

5.5.4.8 Errors

Table 11 — Error Codes for *AddPinhole()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
600-699	TBD	See Control clause in [1].
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.

ErrorCode	errorDescription	Description
703	*InboundPinholeNotAllowed*	Creation of inbound pinholes by UPnP control points 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).

5.5.5 *UpdatePinhole()*

5.5.5.1 Description

This action updates a pinhole's lease time.

5.5.5.2 Arguments

Table 12 — Arguments for *UpdatePinhole()*

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

5.5.5.3 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**.

5.5.5.4 Service Requirements

Before processing the action request, the service shall 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 service shall return the 606 *Action not authorized* error code (defined in [1]). [2] recommends access control requirements and authentication levels to be applied by *default* for this action. However, services may choose a different security policy.

In particular, [2] 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.

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.

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.

5.5.5.5 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 [2] should use their IPv6 GUA when updating a firewall pinhole.

5.5.5.6 Dependency on Service State

FirewallEnabled shall be true indicating that firewall is active.

InboundPinholeAllowed shall be true indicating that UPnP control points can create and update inbound pinhole.

The control point shall have sufficient access rights to update pinhole.

5.5.5.7 Effect on Service State

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

5.5.5.8 Errors

Table 13 — Error Codes for *UpdatePinhole()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
600-699	TBD	See Control clause in [1].
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 control points are not allowed and this action is disabled.
704	*NoSuchEntry*	There is no pinhole with the specified *UniqueID*.

5.5.6 *DeletePinhole()*

5.5.6.1 Description

This action removes a pinhole.

5.5.6.2 Arguments

Table 14 — Arguments for *DeletePinhole()*

Argument	Direction	relatedStateVariable
UniqueID	*IN*	*A_ARG_TYPE_UniqueID*

5.5.6.3 Argument Descriptions

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

5.5.6.4 Service Requirements

Before processing the action request, the service shall 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 service shall return the 606 *Action not authorized* error code (defined in [1]).

[2] recommends access control requirements and authentication levels to be applied by *default* for this action. However, services may choose a different security policy.

In particular, [2] 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.

5.5.6.5 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 [2] should use their IPv6 GUA when deleting a firewall pinhole.

5.5.6.6 Dependency on Service State

FirewallEnabled shall be true indicating that firewall is active.

InboundPinholeAllowed shall be true indicating that UPnP control points can create and delete inbound pinhole.

The control point shall have sufficient access rights to delete pinhole.

5.5.6.7 Effect on Service State

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

5.5.6.8 Errors

Table 15 — Error Codes for *DeletePinhole()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
600-699	TBD	See Control clause in [1].
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 control points are not allowed and this action is disabled.
704	*NoSuchEntry*	There is no pinhole with the specified *UniqueID*.

5.5.7 *GetPinholePackets()*

5.5.7.1 Description

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 control point 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 control point retrieves the *PinholePackets* value at a later time, and the value has increased, the control point 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, another limitation is that some services specify that all established and related network packets should pass automatically the firewall. As a result, with those services, the *PinholePackets* value will not increase even if there is packets going through the service as only the first packet will go through the pinhole.

5.5.7.2 Arguments

Table 16 — Arguments for *GetPinholePackets()*

Argument	Direction	relatedStateVariable
UniqueID	*IN*	*A_ARG_TYPE_UniqueID*
PinholePackets	*OUT*	*A_ARG_TYPE_PinholePackets*

5.5.7.3 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.

5.5.7.4 Service Requirements

Before processing the action request, the service shall 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 service shall return the 606 *Action not authorized* error code (defined in [1]). [2] recommends access control requirements and authentication levels to be applied by *default* for this action. However, services may choose a different security policy.

In particular, [2] 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.

5.5.7.5 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 [2] should use their IPv6 GUA when getting the packets for a firewall pinhole.

5.5.7.6 Dependency on Service State

FirewallEnabled shall be true indicating that firewall is active.

InboundPinholeAllowed shall be true indicating that UPnP control points can create pinhole.

5.5.7.7 Effect on Service State

None.

5.5.7.8 Errors

Table 17 — Error Codes for *GetPinholePackets()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
600-699	TBD	See Control clause in [1].
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 control points are not allowed and this action is disabled.
704	*NoSuchEntry*	There is no pinhole with the specified *UniqueID*.

5.5.8 *CheckPinholeWorking()*

5.5.8.1 Description

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 service may return the 709 *NoTrafficReceived* error code if no traffic corresponding to the specified pinhole was received.

5.5.8.2 Arguments

Table 18 — Arguments for *CheckPinholeWorking()*

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

5.5.8.3 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.

5.5.8.4 Service Requirements

Before processing the action request, the service shall 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 service shall return the 606 *Action not authorized* error code (defined in [1]).

[2] recommends access control requirements and authentication levels to be applied by *default* for this action. However, services may choose a different security policy.

In particular, [2] 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.

5.5.8.5 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 [2] should use their IPv6 GUA when checking a firewall pinhole.

Dependency on Service State

FirewallEnabled shall be true indicating that firewall is active.

InboundPinholeAllowed shall be true indicating that UPnP control points can create pinhole.

5.5.8.6 Effect on Service State

None.

5.5.8.7 Errors

Table 19 — Error Codes for *CheckPinholeWorking()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
600-699	TBD	See Control clause in [1].
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 control points 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.

5.5.9 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.

5.5.10 Error Code Summary

Table 20 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 20 — Error Code Summary

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
600-699	TBD	See Control clause in [1].
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 control points 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 Control clause in [1] for more details.

5.6 Service Behavioral Model

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 [9], 4.3.1, and REC-18, REC-36, REC-41, REC-45 of [6]:

- 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 [9], 4.3.1:

- 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 shall 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 [2] 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 [6].

6 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>
```


Annex A
(informative)

Theory of Operation

A.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 control points 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 remainder of Annex A gives a detailed overview of how to use the services of *WANIPv6FirewallControl*.

A.2 Start-up

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

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

If the firewall is disabled, the UPnP control point:

- 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 control point is not allowed to create inbound pinholes, the UPnP control point:

- 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 control point is allowed to create inbound pinholes, a detailed overview of how to proceed is available in A.4.

A.3 Outbound pinhole management

A.3.1 Outbound pinhole creation

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

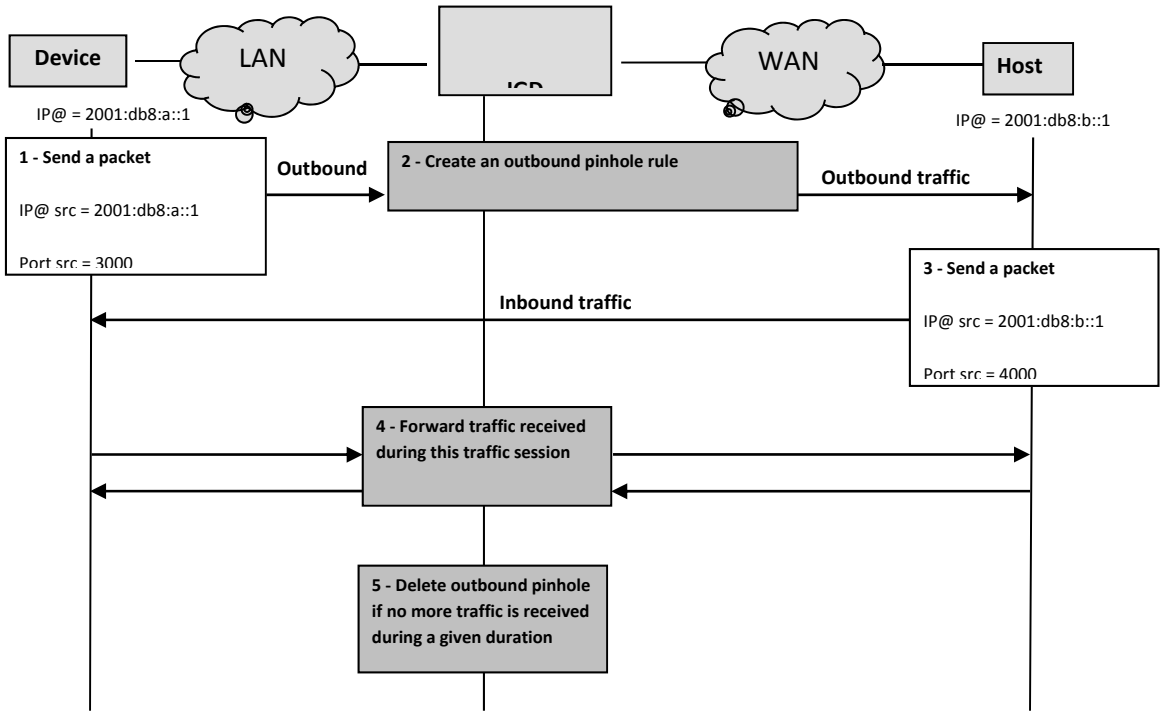

Figure A.1 — Outbound pinhole creation

In Figure A.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 A.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 A.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.

A.3.2 Outbound pinhole refresh

In some cases, the UPnP control point 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 control point can keep the pinhole open by checking the *LeaseTime* and sending a packet before it expires. Conversely, if this control point wants to optimize energy consumption needed to keep this pinhole open, it can use *GetOutboundPinholeTimeout()* action as described in Figure 2.

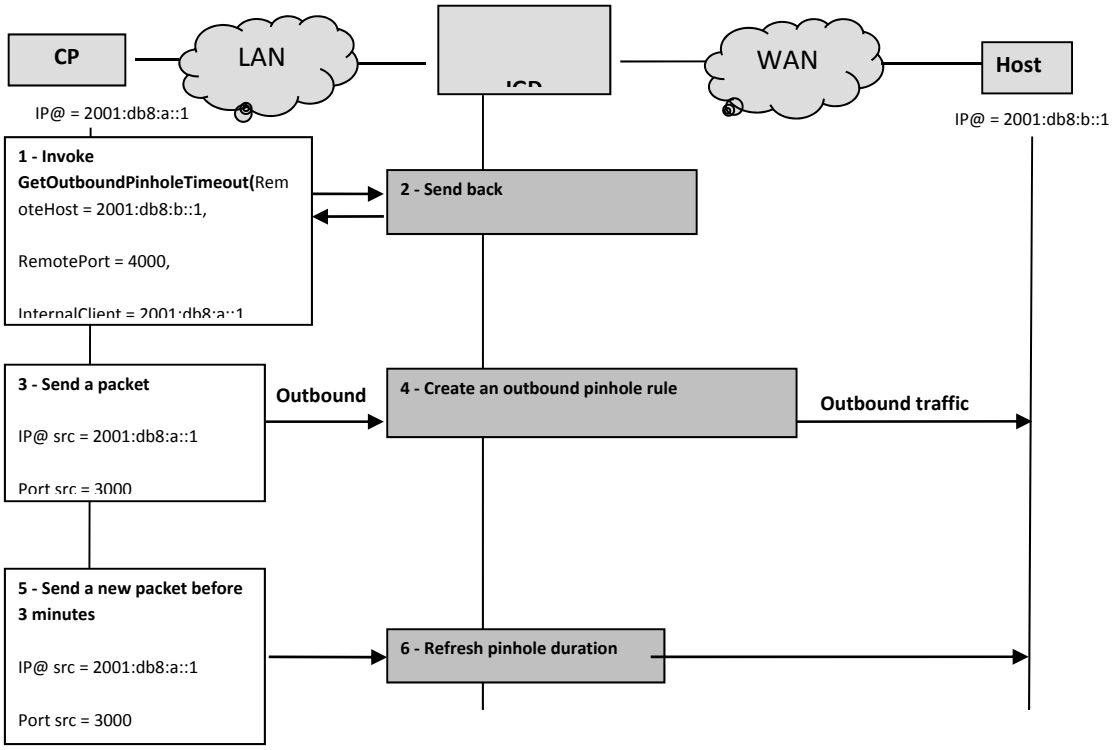

Figure A.2 — Outbound pinhole refresh

A.3.3 Outbound pinhole lifecycle

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

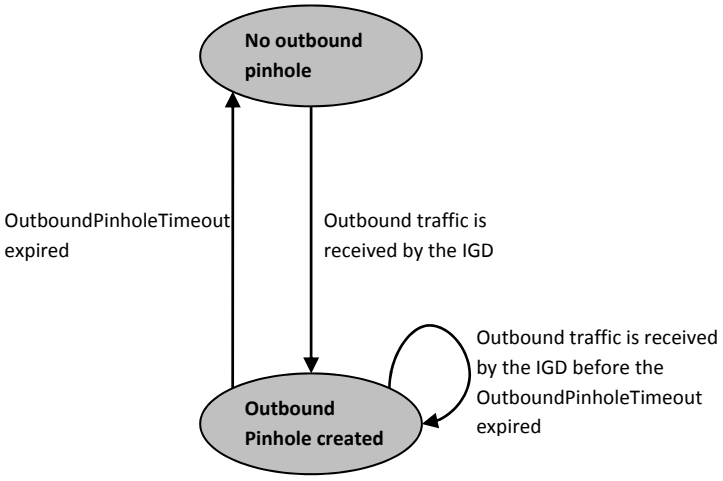

Figure A.3 — Outbound pinhole state transition diagram

A.4 Inbound Pinhole management

A.4.1 Inbound pinhole creation

If a UPnP control point 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 4.

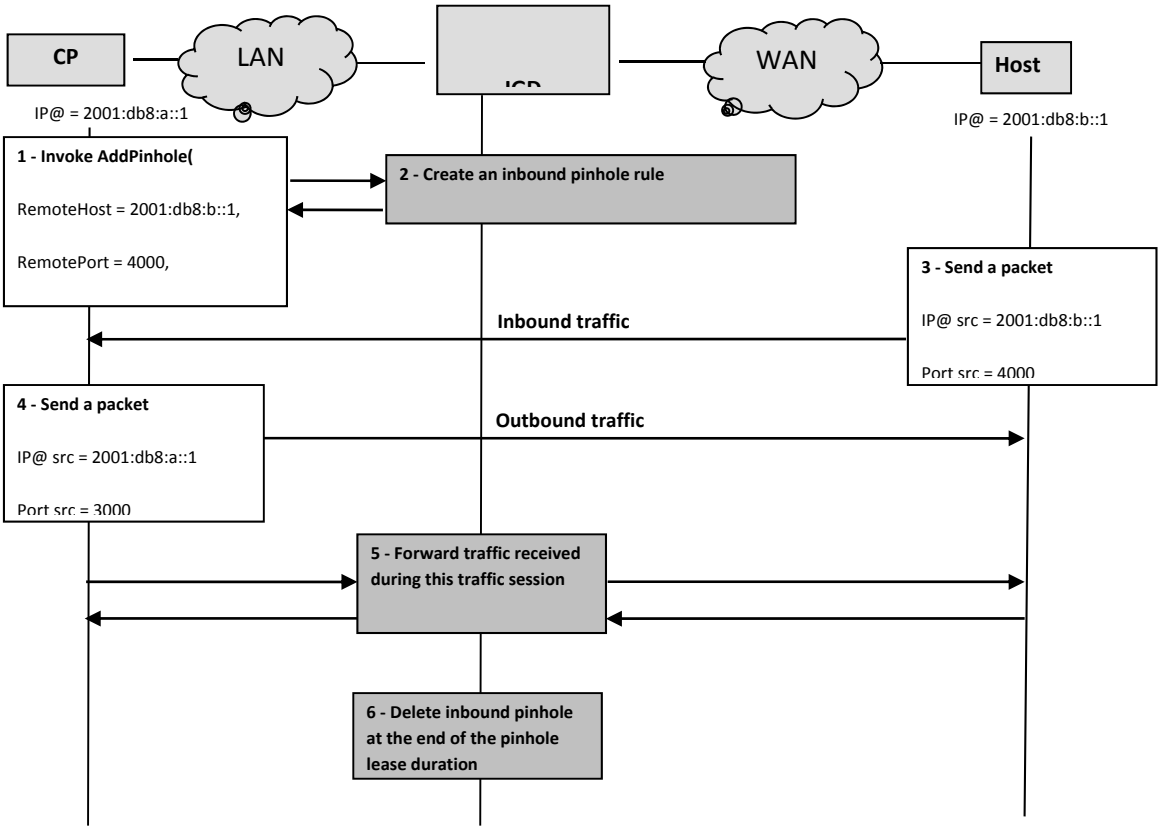

Figure A.4 — Inbound pinhole creation

If the control point 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.

A.4.2 Checking that an inbound pinhole is working

If the UPnP control point 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 A.5. If the *PinholePackets* value returned by this action is different than 0, the UPnP control point can know that its pinhole had worked, at least once. Moreover, if the control point retrieves the *PinholePackets* value at a later time, and the value has increased, the control point can suppose that the pinhole is still working.

If the UPnP control point 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 5.

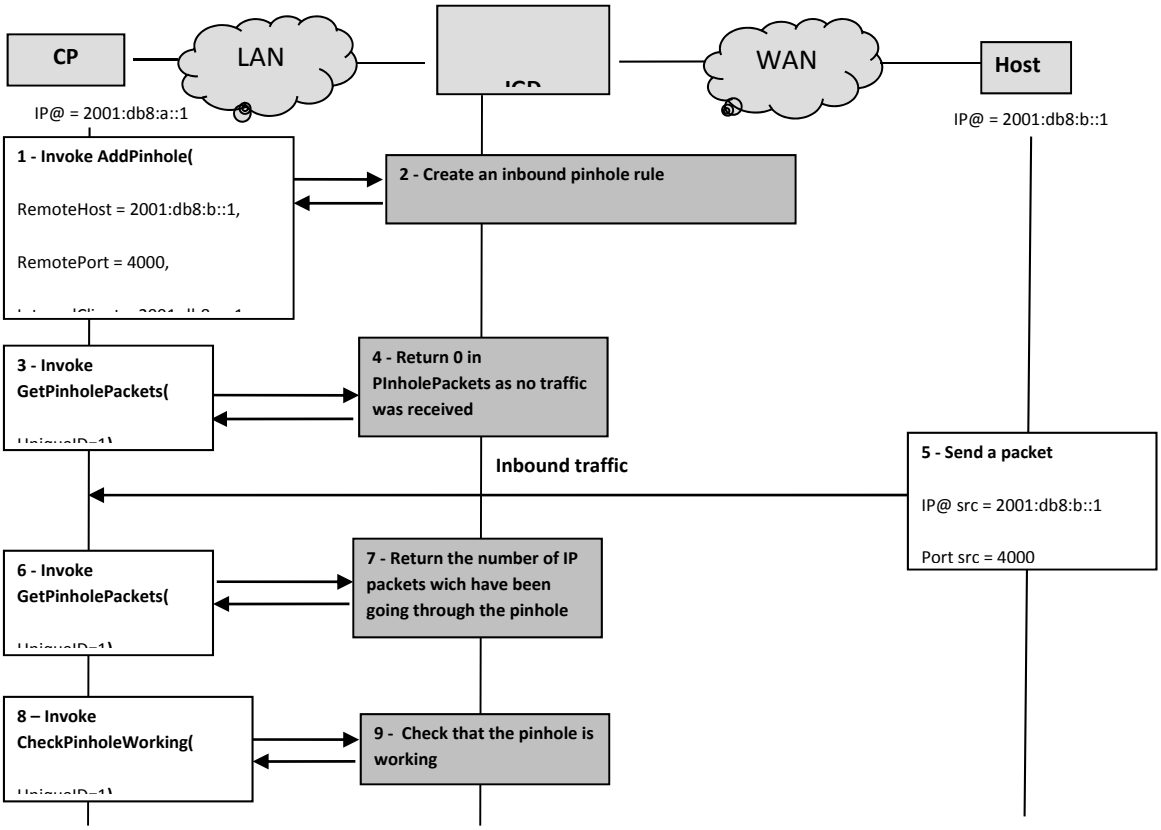

Figure A.5 — Checking that an inbound pinhole is working

A.4.3 Inbound pinhole refresh

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

If the UPnP control point 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 6.

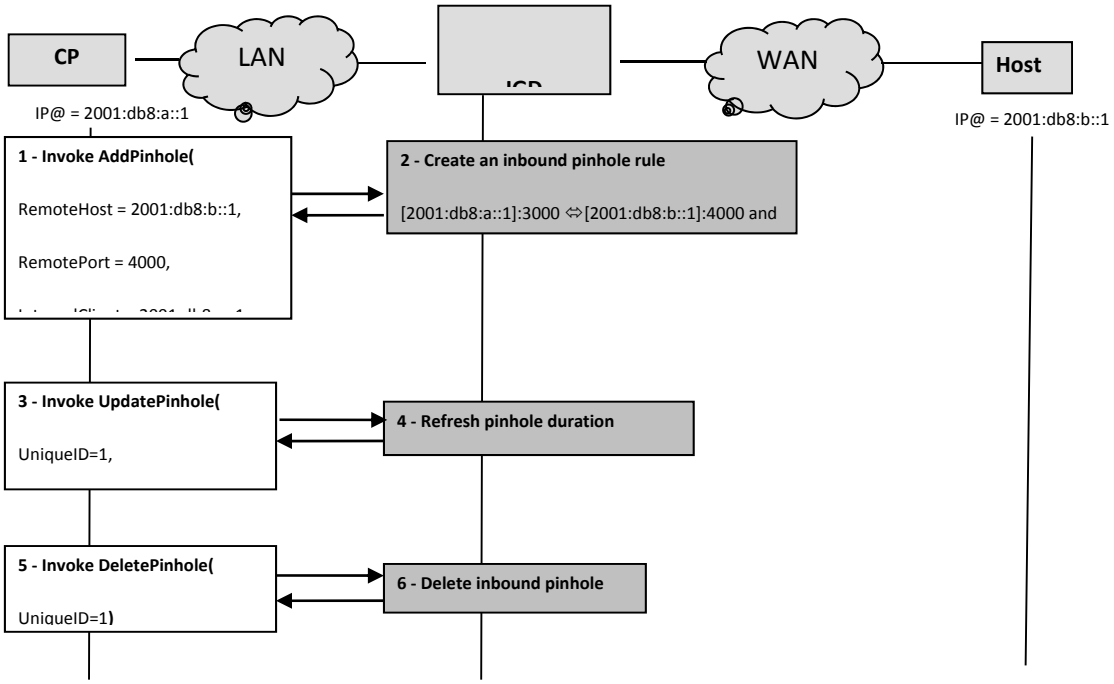

Figure A.6 — Inbound pinhole refresh and deletion

A.4.4 Inbound pinhole state transition diagram

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

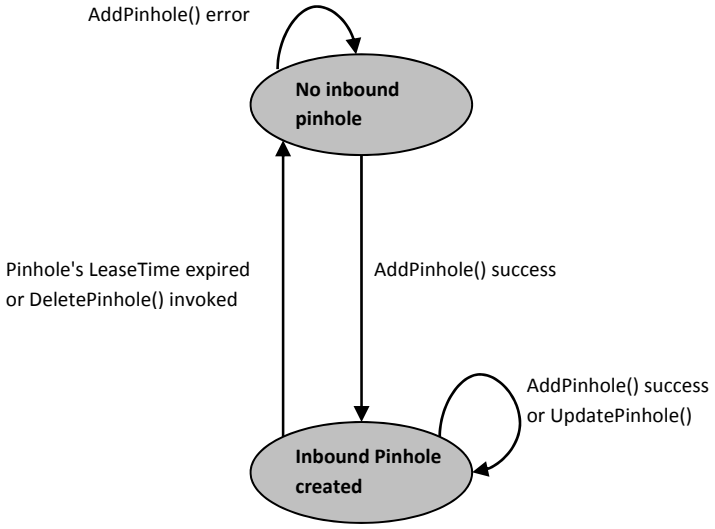

Figure A.7 — Inbound pinhole state transition diagram

Annex B
(normative)

Security Considerations

B.1 Overview

Annex B:

- identifies the threats to the *WANIPv6FirewallControl* and the steps taken in the service to protect against various threats,
- considers the policy alternatives for *WANIPv6FirewallControl*,
- summarizes the recommended policy defaults for this service, 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 [14]. 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, Clause 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.

B.2 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.

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.

B.3 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. Clause B.3 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 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:

h) Whether 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 specification, and this document makes no recommendation.

i) Whether 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.

j) Whether *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.

k) Whether *WANIPv6FirewallControl* actions such as pinhole creation are authenticated by default:

   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. *WANIPv6FirewallControl* should use DeviceProtection with public key cryptography and may also use it with password access control.

l) Whether 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: All implementations of *WANIPv6FirewallControl* should, *by default*, use UPnP *DeviceProtection* for the creation of firewall pinholes. The default configuration of UPnP *DeviceProtection* for this service is explained in (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 by an attacker.

Annex C
(informative)

Bibliography

The following documents, in whole or in part, may be useful for understanding this document but they are not essential for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

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

ICS 35.200

Price based on 31 pages