



BSI Standards Publication

Information technology — UPnP Device Architecture

Part 1-2: UPnP Device Architecture Version 2.0

National foreword

This British Standard is the UK implementation of ISO/IEC 29341-1-2:2017.

The UK participation in its preparation was entrusted to Technical Committee ICT/-/1, Information systems co-ordination.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2017
Published by BSI Standards Limited 2017

ISBN 978 0 580 91063 0

ICS 35.200

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 August 2017.

Amendments/corrigenda issued since publication

Date	Text affected
------	---------------

**Information technology — UPnP
Device Architecture —**

**Part 1-2:
UPnP Device Architecture Version 2.0**

*Technologies de l'information — Architecture de dispositif UPnP —
Partie 1-2: Architecture de dispositif UPnP, version 2.0*



Reference number
ISO/IEC 29341-1-2:2017(E)

Documents
AVA Document Sharing Hub
© ISO/IEC 2017



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2017, Published in Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

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

Introduction	1
0 Addressing	7
0.1 Determining whether to use Auto-IP	7
0.2 Choosing an address	7
0.3 Testing the address	8
0.4 Forwarding rules	8
0.5 Periodic checking for dynamic address availability	9
0.6 Device naming and DNS interaction	9
0.7 Name to IP address resolution	9
0.8 References	9
1 Discovery	10
1.1 SSDP message format	13
1.1.1 SSDP Start-line	13
1.1.2 SSDP message header fields	13
1.1.3 SSDP header field extensions	14
1.1.4 UUID format and recommended generation algorithms	14
1.1.5 SSDP processing rules	14
1.2 Advertisement	15
1.2.1 Advertisement protocols and standards	15
1.2.2 Device available - NOTIFY with ssdp:alive	16
1.2.3 Device unavailable -- NOTIFY with ssdp:byebye	21
1.2.4 Device Update – NOTIFY with ssdp:update	23
1.3 Search	25
1.3.1 Search protocols and standards	25
1.3.2 Search request with M-SEARCH	26
1.3.3 Search response	29
1.4 References	32
2 Description	32
2.1 Generic requirements on HTTP usage	35
2.2 Generic requirements on XML usage	38
2.3 Device description	38
2.4 UPnP Device Template	43
2.5 Service description	44
2.5.1 Defining and processing extended data types	51
2.5.2 String equivalents of extended data types	52
2.5.3 Generic requirements	53
2.5.4 Ordering of Elements	53
2.5.5 Versioning	54
2.6 UPnP Service Template	54
2.7 Non-standard vendor extensions and limitations	54
2.7.1 Placement of Additional Elements and Attributes	56
2.8 UPnP Device Schema	56
2.9 UPnP Service Schema	56
2.10 UPnP Datatype Schema	56
2.11 Retrieving a description using HTTP	56

2.12	References	59
3	Control	60
3.1	Control protocols	63
3.1.1	SOAP Profile	63
3.2	Actions	67
3.2.1	Action invocation	67
3.2.2	Action Response	70
3.2.3	UPnP Action Schema	72
3.2.4	Recommendations and additional requirements	72
3.2.5	Action error response	73
3.2.6	UPnP Error Schema	76
3.3	Query for variable	76
3.4	References	77
4	Eventing	77
4.1	Unicast eventing	78
4.1.1	Subscription	79
4.1.2	SUBSCRIBE with NT and CALLBACK	81
4.1.3	Renewing a subscription with SUBSCRIBE with SID	84
4.1.4	Canceling a subscription with UNSUBSCRIBE	85
4.2	Multicast Eventing	86
4.3	Event messages	88
4.3.1	Error Cases	88
4.3.2	Unicast eventing: Event messages: NOTIFY	89
4.3.3	Multicast Eventing: Event messages: NOTIFY	92
4.4	UPnP Event Schema	95
4.5	Augmenting the UPnP Device and Service Schemas	95
4.6	References	95
5	Presentation	96
5.1	References	97
Annex A	(normative) IP Version 6 Support	98
A.0	Note (informative)	98
A.1	Introduction	98
A.2	General Principles	98
A.2.1	UPnP Device Architecture V1.0	98
A.2.2	UPnP Device Architecture V2.0	99
A.2.3	IPv6 and Dual Stack	99
A.2.4	Device operation	100
A.2.5	Control point operation	101
A.3	Addressing	101
A.3.1	UPnP Messaging on IPv6 Interfaces	101
A.3.2	Summary of boot/startup process	102
A.3.3	Address Selection and RFC 6724	102
A.4	Discovery	102
A.4.1	OPT and NLS	102
A.4.2	Advertisement	103
A.4.3	Advertisement: Device unavailable	103
A.4.4	Advertisement: Device update	104
A.4.5	Search	104

A.4.6	Search response	104
A.5	Description	104
A.6	Control	104
A.7	Eventing	105
A.8	Presentation	105
A.9	References	105
A.9.1	Normative	105
A.9.2	Informative	106
Annex B	Schemas	107
B.1	UPnP Device Schema	107
B.2	UPnP Service Schema	111
B.3	UPnP Control Schema	115
B.4	UPnP Error Schema	116
B.5	UPnP Event Schema	117
B.6	UPnP Cloud Schema	118
B.7	Schema references	119
Annex C	Cloud	120
C.1	Introduction	120
C.1.1	What is UPnP™ Cloud Technology (UCA)?	120
C.1.2	Audience	120
C.1.3	In this Annex	120
C.1.4	UDA compared to UCA	122
C.1.5	UCA General Communications Paths	124
C.1.6	UCA Specific Communication Paths	125
C.1.7	UCA Steps as Analogies to UDA	126
C.2	Terms and Definitions	127
C.2.1	Acronyms	127
C.2.2	General Cloud Terms and Definitions	128
C.2.3	Device and Control Point Terms and Definitions	128
C.2.4	Service Terms and Definitions	129
C.2.5	Groups	129
C.3	References	129
C.4	General XMPP Features	130
C.4.1	XMPP Jabber IDs or <u>JIDs</u>	130
C.5	Creating a Device or Control Point Resource	132
C.5.1	Finding a UCS	132
C.5.2	Account Creation	132
C.5.3	Authentication	133
C.5.4	Binding Devices and Control Points as a Resource	135
C.5.5	Embedded Devices	138
C.6	Presence and Discovery	140
C.6.1	Presence (Analog to NOTIFY with ssdp:alive)	140
C.6.2	XMPP disco#items (analog to M-SEARCH for users UCCDs and UCC-CPs)	144
C.6.3	Presence update (analog to NOTIFY with ssdp:update)	145
C.6.4	Presence "unavailable" (Analog to NOTIFY with ssdp:byebye)	145
C.6.5	Service Level Discovery	146
C.6.6	IQ:Query for DDD and SCPD Exchange (analog of HTTP GET for DDD and SCPD)	146

C.7	<u>PubSub</u> (Analog of Eventing)	155
C.7.1	Creating the UCCD <u>PubSub</u> structure	159
C.7.2	Creating a UCCD PubSub collection	161
C.7.3	Publishing a UCCD PubSub event	166
C.7.4	Subscribing to a UCCD PubSub collection	169
C.7.5	Unsubscribing to a UCCD <u>PubSub</u> collection	171
C.7.6	Permissions model	173
C.8	SOAP over XMPP (Analog of Control)	173
C.9	Support for Binary (Media) Transport	177
C.10	UCA errorCodes	177
C.11	UCA Schemas	178
C.12	Closing a UCA Session	178
C.13	UCA over BOSH and WebSocket	178
Figure 1:	— Protocol stack	1
Figure 1-1:	— Discovery architecture	11
Figure 1-2:	— Advertisement protocol stack	15
Figure 1-3:	— Initial and repeat announcements, no announcement spreading	17
Figure 1-4:	— Initial and repeat announcements, message spreading of repeat announcements	18
Figure 1-5:	— Search protocol stack	25
Figure 2-1:	— Description architecture	33
Figure 2-2:	— Description retrieval protocol stack	57
Figure 3-1:	— Control architecture	61
Figure 3-2:	— Control protocol stack	63
Figure 4-1:	— Unicast eventing architecture	78
Figure 4-2:	— Unicast eventing protocol stack	79
Figure 4-3:	— Multicast eventing architecture	86
Figure 4-4:	— Multicast eventing protocol stack	87
Figure 5-1:	— Presentation architecture	96
Figure 5-2:	— Presentation protocol stack	96
Figure C-1:	— Protocol stacks UDA versus UCA	122
Figure C-2:	— Protocol stack UCA UCCD/UCC-CP and UCA Servers (UCS or UCOD)	123
Figure C-3:	— General UCA Configuration	124
Figure C-4:	— Specific UCA communications	125
Figure C-5:	— XMPP Authentication Negotiation	133
Figure C-6:	— Stanza routing for applications with UCA and other XMPP functionality	138
Figure C-7:	— UDA to UCA Mapping of embedded devices	140
	The individual <u>presence</u> exchange between the UCCDs, UCC-CPs, and UCS for an N connected UPnP scenario is illustrated in	143
Figure C-8:	— Self <presence> stanza flows	144
Figure C-9:	— Combined Connect, Announce and Describe Message Flow	153
Figure C-10:	— <u>PubSub</u> Hierarchy Event Structure Creation	158
Figure C-11:	— BOSH and WebSocket UCA Stack	178
Figure C-12:	— BOSH and WebSocket at UCA component stacks	180

Table 1 — Acronyms	4
Table 1-1 — Root device discovery messages	16
Table 1-2 — Embedded device discovery messages	16
Table 1-3 — Service discovery messages	16
Table 2-1: — Vendor extensions	54
Table 3-1: — SOAP 1.1 UPnP Profile	64
Table 3-2: — mustUnderstand attribute	65
Table 3-3: — UPnP Defined Action error codes	75
Table 4-4: — HTTP Status Codes indicating a Subscription Error	83
Table 4-5: — HTTP Status Codes indicating a Resubscription Error	85
Table 4-6: — HTTP Status Codes indicating a Cancel Subscription Error	86
Table 4-7: — HTTP Status Codes indicating a Notify Error	92
Table 4-8: — Multicast event levels	94
Table A-1: — Matching of Device Address to Multicast Scope	100
Table C-1: — Acronyms	127
Table C-2: — Mapping of DDD iconList to [XEP-0084]	150
Table C-3: — Summary of Requirements for DDD elements	154
Table C-4: — PubSub Node Types	155
Table C-5: — PubSub Node Access Models	155
Table C-6: — PubSub Affiliations and their Privileges to "publishing" as defined by [XEP-0060] and further restricted by UCA (see footnotes)	156
Table C-7: — PubSub Affiliations and their Privileges to "subscribers"	157

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 Standards, 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 29341-1-2 was 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 WANPPPPConnection: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

Introduction

What is UPnP Technology?

UPnP technology defines an architecture for pervasive peer-to-peer network connectivity of intelligent appliances, wireless devices, and PCs of all form factors. It is designed to bring easy-to-use, flexible, standards-based connectivity to ad-hoc or unmanaged networks whether in the home, in a small business, public spaces, or attached to the Internet. UPnP technology provides a distributed, open networking architecture that leverages TCP/IP and Web technologies to enable seamless proximity networking in addition to control and data transfer among networked devices.

The UPnP Device Architecture (UDA) is more than just a simple extension of the plug and play peripheral model. It is designed to support zero-configuration, "invisible" networking, and automatic discovery for a breadth of device categories from a wide range of vendors. This means a device can dynamically join a network, obtain an IP address, convey its capabilities, and learn about the presence and capabilities of other devices. Finally, a device can leave a network smoothly and automatically without leaving any unwanted state behind.

The technologies leveraged in the UPnP architecture include Internet protocols such as IP, TCP, UDP, HTTP, and XML. Like the Internet, contracts are based on wire protocols that are declarative, expressed in XML, and communicated via HTTP. Using Internet protocols is a strong choice for UDA because of its proven ability to span different physical media, to enable real world multiple-vendor interoperability, and to achieve synergy with the Internet and many home and office intranets. The UPnP architecture has been explicitly designed to accommodate these environments. Further, via bridging, UDA accommodates media running non-IP protocols when cost, technology, or legacy prevents the media or devices attached to it from running IP.

What is "universal" about UPnP technology? No device drivers; common protocols are used instead. UPnP networking is media independent. UPnP devices can be implemented using any programming language, and on any operating system. The UPnP architecture does not specify or constrain the design of an API for applications; OS vendors may create APIs that suit their customers' needs.

UPnP Forum

UPnP Forum is an industry initiative designed to enable easy and robust connectivity among stand-alone devices and PCs from many different vendors. UPnP Forum seeks to develop standards for describing device protocols and XML-based device schemas for the purpose of enabling device-to-device interoperability in a scalable, networked environment. For more information, see <http://www.upnp.org>.

UPnP Forum has set up working committees in specific areas of domain expertise. These working committees are charged with creating proposed device standards, building sample implementations, and building appropriate test suites. This document indicates specific technical decisions that are the purview of UPnP Forum working committees.

UPnP vendors can build compliant devices with confidence of interoperability and benefits of shared intellectual property and the logo program. Separate from the logo program, vendors may also build devices that adhere to the UPnP Device Architecture defined herein without a formal standards procedure. If vendors build non-standard devices, they determine technical decisions that would otherwise be determined by a UPnP Forum working committee.

In this document

The UPnP Device Architecture (formerly known as the DCP Framework) contained herein defines the protocols for communication between controllers, or *control points*, and devices. For discovery, description, control, eventing, and presentation, the UPnP Device Architecture uses the following protocol stack (the indicated colors and type styles are used throughout this document to indicate where each protocol element is defined):

Figure 1: — Protocol stack

UPnP vendor [purple-italic]			
UPnP Forum [red-italic]			
UPnP Device Architecture [green-bold]			
SSDP [blue]	Multicast events [navy-bold]	SOAP [blue]	GENA [navy-bold]
		HTTP [black]	HTTP [black]
UDP [black]		TCP [black]	
IP [black]			

At the highest layer, messages logically contain only UPnP vendor-specific information about their devices. Moving down the stack, vendor content is supplemented by information defined by UPnP Forum working committees. Messages from the layers above are hosted in UPnP-specific protocols such as the Simple Service Discovery Protocol (SSDP), the General Event Notification Architecture (GENA) and the multicast event protocol defined in this document, and others that are referenced. SSDP is delivered via either multicast or unicast UDP. Multicast events are delivered via multicast UDP. [GENA](#) is delivered via HTTP. Ultimately, all messages above are delivered over IP. The remaining clauses of this document describe the content and format for each of these protocol layers in detail. For reference, colors in [square brackets] above indicate which protocol defines specific message components throughout this document.

Two general classifications of devices are defined by the UPnP architecture: controlled devices (or simply “devices”), and control points. A controlled device functions in the role of a server, responding to requests from control points. Both control points and controlled devices can be implemented on a variety of platforms including personal computers and embedded systems. Multiple devices, control points, or both may be operational on the same network endpoint simultaneously.

Note: This document is oriented toward an IPv4 environment. Considerations for an IPv6 environment are expressed in Annex A.

The foundation for UPnP networking is IP addressing. In an IPv4 environment, each device or control point shall have a Dynamic Host Configuration Protocol (DHCP) client and search for a DHCP server when the device or control point is first connected to the network. If a DHCP server is available, i.e., the network is managed; the device or control point shall use the IP address assigned to it. If no DHCP server is available, i.e., the network is unmanaged; the device or control point shall use Auto IP to get an address. In brief, Auto IP defines how a device or control point intelligently chooses an IP address from a set of reserved addresses and is able to move easily between managed and unmanaged networks. If during the DHCP transaction, the device or control point obtains a domain name, e.g., through a DNS server or via DNS forwarding, the device or control point should use that name in subsequent network operations; otherwise, the device or control point should use its IP address.

Certain UPnP networks have more complex configurations such as multiple physical networks and/or multiple logical networks to accommodate multiple non-overlapping addressing schemes. Devices and control points may also have two or more network interfaces, and/or two or more IP addresses assigned to each interface. In such configurations, a single device or control point may be assigned multiple IP addresses from different logical networks in the same UPnP network, resulting in devices appearing to a control point multiple times in the network. Devices and control points that have multiple IP addresses on the same UPnP network are referred to as multi-homed. Throughout this document, the term “UPnP-enabled interface” is used to refer to an interface which is assigned an IP address belonging to the UPnP network. Additional behaviors specific to multi-homed devices and control points will be covered in applicable clauses throughout the document. However, as a general principle, related interactions between control points and devices (e.g. action control request and response messages, event subscription and event messages) shall occur using the same pair of outgoing and incoming UPnP-enabled interfaces.

Given an IP address, Step 1 in UPnP networking is *discovery*. When a device is added to the network, the UPnP discovery protocol allows that device to advertise its services to control points on the network. Similarly, when a control point is added to the network, the UPnP discovery protocol allows that control point to search for devices of interest on the network. The fundamental exchange in both cases is a discovery message containing a few essential specifics about the device or one of its services, e.g., its type, identifier, and a pointer to more detailed information. The clause on Discovery below explains how devices advertise, how control points search, and contains details about the format of discovery messages.

Step 2 in UPnP networking is *description*. After a control point has discovered a device, the control point still knows very little about the device. For the control point to learn more about the device and its capabilities, or to interact with the device, the control point shall retrieve the device's description from the URL provided by the device in the discovery message. Devices may contain other logical devices, as well as functional units, or *services*. The UPnP description for a device is expressed in XML and includes vendor-specific manufacturer information like the model name and number, the serial number, the manufacturer name, URLs to vendor-specific Web sites, etc. The description also includes a list of any embedded devices or services, as well as URLs for control, eventing, and presentation. For each service, the description includes a list of the commands, or *actions*, to which the service responds, and parameters, or *arguments* for each action; the description for a service also includes a list of variables; these variables model the state of the service at run time, and are described in terms of their data type, range, and event characteristics. The clause on Description below explains how devices are described and how control points retrieve those descriptions.

Step 3 in UPnP networking is *control*. After a control point has retrieved a description of the device, the control point can send actions to a device's services. To do this, a control point sends a suitable control message to the control URL for the service (provided in the device description). Control messages are also expressed in XML using the Simple Object Access Protocol (SOAP). Like function calls, in response to the control message, the service returns any action-specific values. The effects of the action, if any, are modeled by changes in the variables that describe the run-time state of the service. The clause on Control below explains the description of actions, state variables, and the format of control messages.

Step 4 in UPnP networking is *eventing*. A UPnP description for a service includes a list of actions the service responds to and a list of variables that model the state of the service at run time. The service publishes updates when these variables change, and a control point may subscribe to receive this information. The service publishes updates by sending event messages. Event messages contain the names of one or more state variables and the current value of those variables. These messages are also expressed in XML. A special initial event message is sent when a control point first subscribes; this event message contains the names and values for all evented variables and allows the subscriber to initialize its model of the state of the service. To support scenarios with multiple control points, eventing is designed to keep all control points equally informed about the effects of any action. Therefore, all subscribers are sent all event messages, subscribers receive event messages for all evented variables that have changed, and event messages are sent no matter why the state variable changed (either in response to a requested action or because the state the service is modeling changed). Multicast eventing is a variant of Step 4 in UPnP networking. Through multicast eventing, control points can listen to state changes in services without subscription. This form of eventing is useful first when events which are not relevant to specific UPnP interactions should be delivered to control points to inform users, and second when multiple controlled devices want to inform multiple other control points. Multicast eventing is inherently unreliable since it is based on UDP. To increase the probability of successful transmission, the option to retransmit multicast event notifications is outlined. UPnP Working committees should define whether specific events are multicast events. The clause on Eventing below explains unicast event subscription and the format of both unicast and multicast event messages.

Step 5 in UPnP networking is *presentation*. If a device has a URL for presentation, then the control point can retrieve a page from this URL, load the page into a browser, and depending on the capabilities of the page, allow a user to control the device and/or view device status. The degree to which each of these can be accomplished depends on the specific capabilities

of the presentation page and device. The clause on Presentation below explains the protocol for retrieving a presentation page.

Audience

The audience for this document includes UPnP device and control point vendors, members of UPnP Forum working committees, and anyone else who has a need to understanding the technical details of UPnP protocols.

This document assumes the reader is familiar with the HTTP, TCP, UDP, IP family of protocols; this document makes no attempt to explain them. This document also assumes most readers will be new to XML, and while it is not an XML tutorial, XML-related issues are addressed in detail given the centrality of XML to the UPnP Device Architecture. This document makes no assumptions about the reader's understanding of various programming or scripting languages.

Conformance terminology

In this document, features are described as required, recommended, allowed or DEPRECATED as follows:

Required (or shall or mandatory).

These basic features shall be implemented to comply with UPnP Device Architecture. The phrases “shall not”, and “PROHIBITED” indicate behavior that is prohibited, i.e. that if performed means the implementation is not in compliance.

Recommended (or should).

These features add functionality supported by UPnP Device Architecture and should be implemented. Recommended features take advantage of the capabilities UPnP Device Architecture, usually without imposing major cost increases. Notice that for compliance testing, if a recommended feature is implemented, it shall meet the specified requirements to be in compliance with these guidelines. Some recommended features could become requirements in the future. The phrase “should not” indicates behavior that is permitted but not recommended.

Allowed (or may).

These features are neither required nor recommended by UPnP Device Architecture, but if the feature is implemented, it shall meet the specified requirements to be in compliance with these guidelines. These features are not likely to become requirements in the future.

DEPRECATED.

Although these features are still described in this specification, they should not be implemented except for backward compatibility. The occurrence of a deprecated feature during operation of an implementation compliant with the current specification has no effect on the implementation's operation and does not produce any error conditions. Backward compatibility may require that a feature is implemented and functions as specified but it shall never be used by implementations compliant with this specification.

Acronyms

Table 1 — Acronyms

Acronym	Meaning	Acronym	Meaning
ARP	Address Resolution Protocol	SOAP	Simple Object Access Protocol
CP	Control Point	SSDP	Simple Service Discovery Protocol
DCP	Device Control Protocol	UDA	UPnP Device Architecture
DDD	Device Description Document	UPC	Universal Product Code
DHCP	Dynamic Host Configuration Protocol	URI	Uniform Resource Identifier

DNS	Domain Name System	URL	Uniform Resource Locator
GENA	General Event Notification Architecture	URN	Uniform Resource Name
HTML	Hypertext Markup Language	UUID	Universally Unique Identifier
HTTP	Hypertext Transfer Protocol	XML	Extensible Markup Language
SCPD	Service Control Protocol Description		

Glossary

action

Command exposed by a service. Takes one or more input or output arguments. May have a return value. For more information, see clause 2, “Description” and clause 3, “Control”.

argument

Parameter for action exposed by a service. May be in or out. For more information, see clause 2, “Description” and clause 3, “Control”.

control point

Retrieves device and service descriptions, sends actions to services, polls for service state variables, and receives events from services.

device

Logical device. A container. May embed other logical devices. Embeds one or more services. Advertises its presence on network(s). For more information, see clause 1, “Discovery” and clause 2, “Description”.

device description

Formal definition of a logical device, expressed in the UPnP Template Language. Written in XML syntax. Specified by a UPnP vendor by filling in the placeholders in a UPnP Device Template, including, e.g., manufacturer name, model name, model number, serial number, and URLs for control, eventing, and presentation. For more information, see clause 2, “Description”.

device type

Standard device types are denoted by urn:schemas-upnp-org:device: followed by a unique name assigned by a UPnP Forum working committee. One-to-one relationship with UPnP Device Templates. UPnP vendors may specify additional device types; these are denoted by urn:domain-name:device: followed by a unique name assigned by the vendor, where *domain-name* is a Vendor Domain Name. For more information, see clause 2, “Description”.

event

Notification of one or more changes in state variables exposed by a service. For more information, see clause 4, “Eventing”.

GENA

General Event Notification Architecture. The event subscription and notification protocol defined in clause 4, “Eventing”.

publisher

Source of event messages. Typically a device's service. For more information, see clause 4, “Eventing”.

root device

A logical device that is not embedded in any other logical device. For more information, see clause 2, “Description”.

service

Logical functional unit. Smallest units of control. Exposes actions and models the state of a physical device with state variables. For more information, see clause 3, “Control”.

service description

Formal definition of a logical service, expressed in the UPnP Template language. Written in XML syntax. Specified by a UPnP vendor by filling in any placeholders in a UPnP Service Template. (Was SCPD.) For more information, see clause 2, "Description".

service type

Standard service types are denoted by urn:schemas-upnp-org:service: followed by a unique name assigned by a UPnP forum working committee, colon, and an integer version number. One-to-one relationship with UPnP Service Templates. UPnP vendors may specify additional services; these are denoted by urn:*domain-name*:service: followed by a unique name assigned by the vendor, colon, and a version number, where *domain-name* is a Vendor Domain Name. For more information, see clause 2, "Description".

SOAP

Simple Object Access Protocol. A remote-procedure call mechanism based on XML that sends commands and receives values over HTTP. For more information, see clause 3, "Control".

SSDP

Simple Service Discovery Protocol. A multicast discovery and search mechanism that uses a multicast variant of HTTP over UDP. Defined in clause 1, "Discovery".

state variable

Single facet of a model of a physical service. Exposed by a service. Has a name, data type, optional default value, optional constraints values, and may trigger events when its value changes. For more information, see clause 2, "Description" and clause 3, "Control".

subscriber

Recipient of event messages. Typically a control point. For more information, see clause 4, "Eventing".

UPnP Device Template

Template listing device type, required embedded devices (if any), and required services. Written in XML syntax and derived from the UPnP Device Schema. Defined by a UPnP Forum working committee. One-to-one relationship with standard device types. For more information, see clause 2, "Description".

UPnP Service Template

Template listing action names, parameters for those actions, state variables, and properties of those state variables. Written in XML syntax and derived from the UPnP Service Schema. Defined by a UPnP Forum working committee. One-to-one relationship with standard service types. For more information, see clause 2, "Description".

UPnP Device Schema

Defines the elements and attributes used in UPnP Device and Service Templates. Written in XML syntax and derived from XML Schema (Part 1: Structures, Part 2: Datatypes). Defined by the UPnP Device Architecture herein. For more information, see clause 2, "Description".

Vendor Domain Name

A domain name that is supplied by a vendor. It is owned by the vendor, and shall be registered with an ICANN accredited Registrar, such that it is unique. The vendor shall keep the domain name registration up to date. A Vendor Domain Name length should be chosen to be compatible with the use of the domain name in the UDA.

References and resources

RFC 2710, Multicast Listener Discovery for IPv6. Available at: <http://www.ietf.org/rfc/rfc2710.txt>.

RFC 2616, HTTP: Hypertext Transfer Protocol 1.1. Available at: <http://www.ietf.org/rfc/rfc2616.txt>.

RFC 2279, UTF-8, a transformation format of ISO 10646 (character encoding). Available at: <http://www.ietf.org/rfc/rfc2279.txt>.

XML, Extensible Markup Language. W3C recommendation. Available at: <http://www.w3.org/XML/>.

DEVICEPROTECTION, UPnP Device Protection specification. Available at <http://upnp.org/specs/gw/UPnP-gw-DeviceProtection-v1-Service.pdf>.

Each clause in this document contains additional information about resources for specific topics.

0 Addressing

Addressing is Step 0 of UPnP networking. Through addressing, devices and control points get a network address. Addressing enables discovery (Step 1) where control points find interesting device(s), description (Step 2) where control points learn about device capabilities, control (Step 3) where a control point sends commands to device(s), eventing (Step 4) where control points listen to state changes in device(s), and presentation (Step 5) where control points display a user interface for device(s).

The foundation for UPnP networking is IP addressing. A UPnP device or control point is allowed to support IP version 4-only, or both IP version 4 and IP version 6. This clause, and the examples given throughout clauses 1 through 5 of this document, assumes an IPv4 implementation. Annex A of this document describes IPv6 operation. Each UPnP device or control point which does not itself implement a DHCP server shall have a Dynamic Host Configuration Protocol (DHCP) client and search for a DHCP server when the device or control point is first connected to the network (if the device or control point itself implements a DHCP server, it allowed to allocate itself an address from the pool that it controls). If a DHCP server is available, i.e., the network is managed; the device or control point shall use the IP address assigned to it. If no DHCP server is available, i.e., the network is unmanaged; the device or control point shall use automatic IP addressing (Auto-IP) to obtain an address.

Auto-IP (defined in RFC 3927) defines how a device or control point: (a) determines if DHCP is unavailable, and (b) intelligently chooses an IP address from a set of link-local IP addresses. This method of address assignment enables a device or control point to easily move between managed and unmanaged networks.

This clause provides an overview of the basic operation of Auto-IP. The operations described in this clause are detailed and clarified in the reference documents listed below. Where conflicts between this document and the reference documents exist, the reference document always takes precedence.

0.1 Determining whether to use Auto-IP

A device or control point that supports Auto-IP and is configured for dynamic address assignment begins by requesting an IP address via DHCP by sending out a DHCPDISCOVER message. The amount of time this DHCP Client listens for DHCPOFFERS is implementation dependent. If a DHCPOFFER is received during this time, the device or control point shall continue the process of dynamic address assignment. If no valid DHCPOFFERS are received, the device or control point shall then auto-configure an IP address using Auto-IP.

0.2 Choosing an address

To auto-configure an IP address using Auto-IP, the device or control point uses an implementation dependent algorithm for choosing an address in the 169.254/16 range. The first and last 256 addresses in this range are reserved and shall NOT be used.

The selected address shall then be tested to determine if the address is already in use. If the address is in use by another device or control point, another address shall be chosen and tested, up to an implementation dependent number of retries. The address selection shall be randomized to avoid collision when multiple devices or control points are attempting to allocate addresses. The device or control point chooses an address using a pseudo-random

algorithm (distributed over the entire address range from 169.254.1.0 to 169.254.254.255) to minimize the likelihood that devices or control points that join the network at the same time will choose the same address and subsequently choose alternative addresses in the same sequence when collisions are detected. This pseudo-random algorithm should be seeded using the device's or control point's Ethernet hardware MAC address.

0.3 Testing the address

To test the chosen address, the device or control point shall use an Address Resolution Protocol (ARP) probe. An ARP probe is an ARP request with the device or control point hardware address used as the sender's hardware address and the sender's IP address set to 0s. The device or control point shall then listen for responses to the ARP probe, or other ARP probes for the same IP address. If either of these ARP packets is seen, the device or control point shall consider the address in use and try a different address. The ARP probe is allowed to be repeated for greater certainty that the address is not already in use; it is recommended that the probe be sent four times at two-second intervals.

After successfully configuring a link-local address, the device or control point shall send two gratuitous ARPs, spaced two seconds apart, this time filling in the sender IP address. The purpose of these gratuitous ARPs is to make sure that other hosts on the net do not have stale ARP cache entries left over from some other host that may previously have been using the same address.

Devices and control points that are equipped with persistent storage are allowed to record the IP address they have selected and on the next boot use that address as their first candidate when probing, in order to increase the stability of addresses and reduce the need to resolve address conflicts.

Address collision detection is not limited to the address testing phase, when the device or control point is sending ARP probes and listening for replies. Address collision detection is an ongoing process that is in effect for as long as the device or control point is using a link-local address. At any time, if a device or control point receives an ARP packet with its own IP address given as the sender IP address, but a sender hardware address that does not match its own hardware address, then the device or control point shall treat this as an address collision and shall respond as described in either a) or b) below:

- a) Immediately configure a new link-local IP address as described above; or,
- b) If the device or control point currently has active TCP connections or other reasons to prefer to keep the same IP address, and has not seen any other conflicting ARP packets recently (e.g., within the last ten seconds) then it is allowed to elect to attempt to defend its address once, by recording the time that the conflicting ARP packet was received, and then broadcasting one single gratuitous ARP, giving its own IP and hardware addresses as the source addresses of the ARP. However, if another conflicting ARP packet is received within a short time after that (e.g., within ten seconds) then the device or control point shall immediately configure a new Auto-IP address as described above.

The device or control point shall respond to conflicting ARP packets as described in either a) or b) above; it shall NOT ignore conflicting ARP packets. If a new address is selected, the device or control point shall cancel previous advertisements and re-advertise with the new address.

After successfully configuring an Auto-IP address, all subsequent ARP packets (replies as well as requests) containing an Auto-IP source address shall be sent using link-level *broadcast* instead of link-level *unicast*, in order to facilitate timely detection of duplicate addresses.

0.4 Forwarding rules

IP packets whose source or destination addresses are in the 169.254/16 range shall NOT be sent to any router for forwarding. Instead, the senders shall ARP for the destination address and then send the packets directly to the destination on the same link. IP datagrams with a multicast destination address and an Auto-IP source address shall NOT be forwarded off the local link. Devices and control points are allowed to assume that all 169.254/16 destination

addresses are on-link and directly reachable. The 169.254/16 address range shall not be subnetted.

0.5 Periodic checking for dynamic address availability

A device or control point that has auto-configured an IP address shall periodically check for the existence of a DHCP server. This is accomplished by sending DHCPDISCOVER messages. How often this check is made is implementation dependent, but checking every 5 minutes would maintain a balance between network bandwidth required and connectivity maintenance. If a DHCPOFFER is received, the device or control point shall proceed with dynamic address allocation. Once a DHCP assigned address is in place, the device or control point is allowed to release the auto-configured address, but is also allowed to choose to maintain this address for a period of time (or indefinitely) to maintain connectivity.

To switch over from one IP address to a new one, the device should, if possible, cancel any outstanding advertisements made on the previous address and shall issue new advertisements on the new address. The clause on Discovery explains advertisements and their cancellations. In addition, any event subscriptions are deleted by the device (see clause on Eventing).

For a multi-homed device with multiple IP addresses, to switch one of the IP addresses to a new one, the device should cancel any outstanding advertisements made on the previous IP address, and shall issue new advertisements on the new IP addresses. Furthermore, it shall also issue appropriate update advertisements on all unaffected IP addresses. The clause on Discovery explains advertisements, their cancellations and updates. The clause on Eventing explains the effect on event subscriptions.

0.6 Device naming and DNS interaction

Once a device has a valid IP address for the network, it can be located and referenced on that network through that address. There may be situations where the end user needs to locate and identify a device. In these situations, a friendly name for the device is much easier for a human to use than an IP address. If a device chooses to provide a host name to a DHCP server and register with a DNS server, the device should either ensure the requested host name is unique or provide a means for the user to change the requested host name. Most often, devices do not provide a host name, but provide URLs using literal (numeric) IP addresses.

Moreover, names are much more static than IP addresses. Clients referring a device by name don't require any modification when the IP address of a device changes. Mapping of the device's DNS name to its IP address could be entered into the DNS database manually or dynamically according to RFC 2136. While devices supporting dynamic DNS updates can register their DNS records directly in the DNS, it is also possible to configure a DHCP server to register DNS records on behalf of these DHCP clients.

0.7 Name to IP address resolution

A device that needs to contact another device identified by a DNS name needs to discover its IP address. The device submits a DNS query according to RFC1034 and 1035 to the pre-configured DNS server(s) and receives a response from a DNS server containing the IP address of the target device. A device can be statically pre-configured with the list of DNS servers. Alternatively a device could be configured with the list of DNS server through DHCP, or after the address assignment through a DHCPINFORM message.

0.8 References

RFC1034, Domain Names - Concepts and Facilities. Available at: <http://www.ietf.org/rfc/rfc1034.txt>.

RFC1035, Domain Names - Implementation and Specification. Available at: <http://www.ietf.org/rfc/rfc1035.txt>.

RFC 2131, Dynamic Host Configuration Protocol. Available at: <http://www.ietf.org/rfc/rfc2131.txt>.

RFC 2136, Dynamic Updates in the Domain Name System. Available at: <http://www.ietf.org/rfc/rfc2136.txt>.

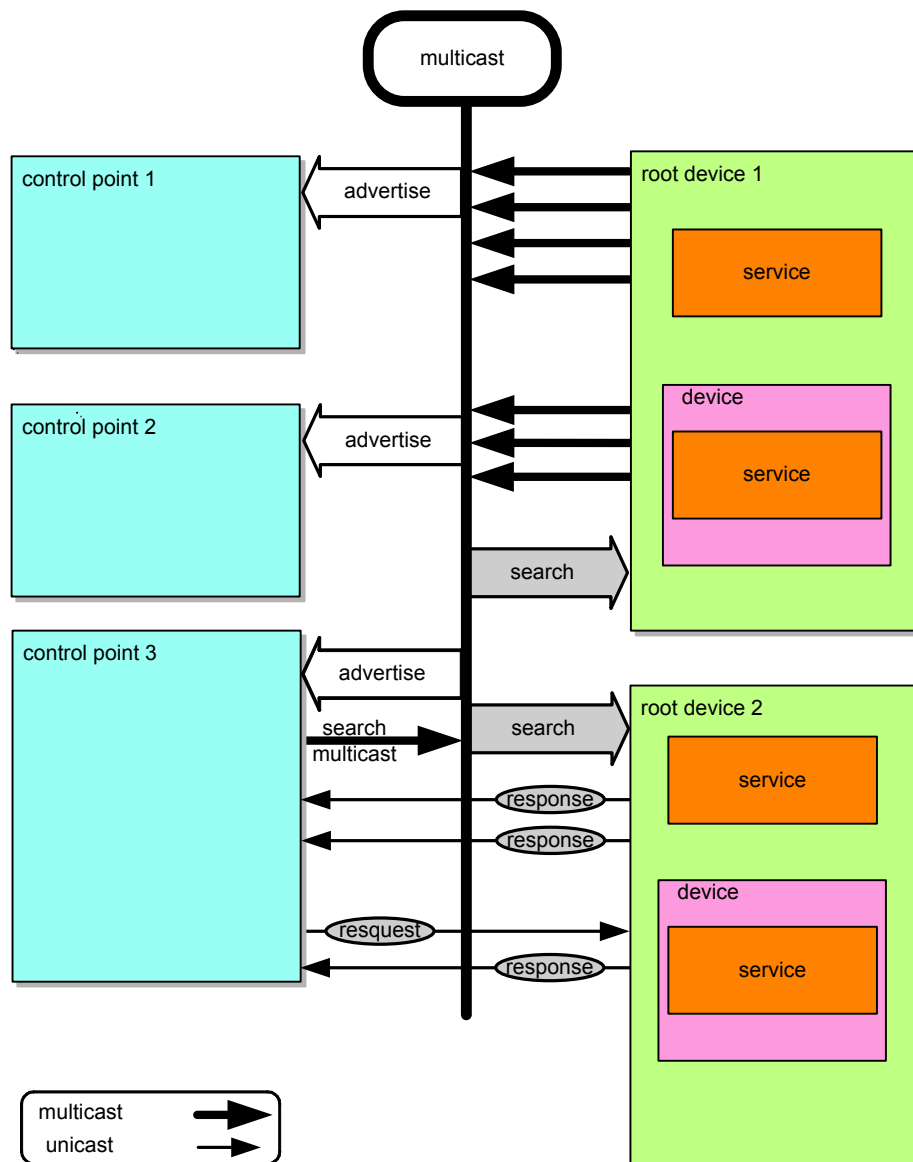
RFC 3927, Dynamic Configuration of IPv4 Link-Local Addresses. Available at: <http://www.ietf.org/rfc/rfc3927.txt>.

1 Discovery

Discovery is Step 1 in UPnP™ networking. Discovery comes after addressing (Step 0) where devices get a network address. Through discovery, control points find interesting device(s). Discovery enables description (Step 2) where control points learn about device capabilities, control (Step 3) where a control point sends commands to device(s), eventing (Step 4) where control points listen to state changes in device(s), and presentation (Step 5) where control points display a user interface for device(s).

Discovery is the first step in UPnP networking. When a device is added to the network, the UPnP discovery protocol allows that device to advertise its services to control points on the network. Similarly, when a control point is added to the network, the UPnP discovery protocol allows that control point to search for devices of interest on the network. The fundamental exchange in both cases is a discovery message containing a few, essential specifics about the device or one of its services, e.g., its type, universally unique identifier, a pointer to more detailed information and optionally parameters that identify the current state of the device.

Figure 1-1: — Discovery architecture



When a device knows it is newly added to the network, it shall multicast a number of discovery messages advertising itself, its embedded devices, and its services (initial announce). Any interested control point can listen to the standard multicast address for notifications that new capabilities are available. A multi-homed device shall multicast the discovery messages on all UPnP-enabled interfaces. A multi-homed control point is allowed to listen to the standard multicast address on one, some or all of its UPnP-enabled interfaces.

When a new control point is added to the network, it is allowed to multicast a discovery message searching for interesting devices, services, or both. All devices shall listen to the standard multicast address for these messages and shall respond if any of their root devices, embedded devices or services matches the search criteria in the discovery message. In addition, a control point is allowed to unicast a discovery message to a specific IP address on port 1900 or on the port specified by the optional SEARCHPORT.UPNP.ORG header field (which supersedes port 1900 for this use), searching for a UPnP device or service at that specific IP address. This action presumes the control point already knows the device at this IP address is a UPnP device (which listens on the appropriate port). The control point can use unicast search for a number of applications. A unicast search can quickly confirm a specific device and provide the corresponding discovery information (e.g. UUID, URL) of this device. All devices shall listen to incoming unicast search messages on port 1900 or, if provided, the port number specified in the SEARCHPORT.UPNP.ORG header field and shall respond if any

of their root devices, embedded devices or services matches the search criteria in the discovery message.

A multi-homed control point is allowed to multicast discovery messages on one, some or all of its UPnP-enabled interfaces. Multi-homed devices shall listen to the standard multicast address on all UPnP-enabled interfaces for multicast discovery messages. Multi-homed devices shall also listen to incoming unicast search messages on port 1900 or, if provided, the port number specified in the SEARCHPORT.UPNP.ORG header field. The devices shall respond if any of their root devices, embedded devices or services matches the search criteria in the discovery message.

To reiterate, a control point is allowed to learn of a device of interest because that device sent discovery messages advertising itself or because the device responded to a discovery message searching for devices. In either case, if a control point is interested in a device and wants to learn more about it, the control point uses the information in the discovery message to send a *description* query message. The clause on Description explains description messages in detail.

When a device is removed from the network, it should, if possible, multicast a number of discovery messages revoking its earlier announcements, effectively declaring that its root devices, embedded devices and services will no longer be available. When the IP address of a device is changed, it should revoke any earlier announcements and it shall advertise using the new IP address.

When a multi-homed device becomes unavailable to the network on any of its UPnP-enabled interfaces, it should, if possible, multicast a number of discovery messages revoking its earlier announcements on the affected UPnP-enabled interfaces, effectively declaring that its root devices, embedded devices and services will no longer be available on those interfaces. If it remains available to the network on any of its other UPnP-enabled interfaces, it shall NOT multicast such discovery messages on the unaffected UPnP-enabled interfaces.

When a multi-homed device becomes available to the network on a new UPnP-enabled interface (in addition to any existing UPnP-enabled interfaces), it shall increase its BOOTID.UPNP.ORG field value (see clause 1.2 “Advertisement”), and multicast a number of update messages on the existing UPnP-enabled interfaces to announce the new BOOTID.UPNP.ORG field value. After all the update messages have been sent, it shall multicast a number of discovery messages on all (existing and new) UPnP-enabled interfaces with the new BOOTID.UPNP.ORG field value.

Similarly, when one of the IP addresses of a multi-homed device is changed, it should revoke any earlier announcements on the previous IP address. It shall increase its BOOTID.UPNP.ORG field value (see clause 1.2 “Advertisement”), and multicast a number of update messages on the existing UPnP-enabled interfaces to announce the new BOOTID.UPNP.ORG field value. After all the update messages have been sent, it shall multicast a number of discovery messages on all (existing and new) UPnP-enabled interfaces with the new BOOTID.UPNP.ORG field value.

Finally, if a multi-homed device loses connectivity on one of its UPnP-enabled interfaces and then regains connectivity, it shall increase its BOOTID.UPNP.ORG field value (see 1.2, “Advertisement”), and multicast a number of update messages on the unaffected UPnP-enabled interfaces to announce the new BOOTID.UPNP.ORG field value. After all the update messages have been sent, it shall multicast a number of discovery messages on all (affected and unaffected) UPnP-enabled interfaces with the new BOOTID.UPNP.ORG field value.

To limit network congestion, the time-to-live (TTL) of each IP packet for each multicast message should default to 2 and should be configurable. When the TTL is greater than 1, it is possible for multicast messages to traverse multiple routers; therefore control points and devices using non-AutoIP addresses shall send an IGMP Join message so that routers will forward multicast messages to them (this is not necessary when using an Auto-IP address, since packets with Auto-IP addresses will not be forwarded by routers).

Versioning: Discovery plays an important role in the interoperability of devices and control points using different versions of UPnP networking. The UPnP Device Architecture (defined herein) is versioned with both a major and a minor version, usually written as *major.minor*, where both *major* and *minor* are integers (for example, version 2.10 [two dot ten] is *newer* than version 2.2 [two dot two]). Advances in minor versions shall be a compatible superset of earlier minor versions of the same major version. Advances in major version are not required to be supersets of earlier versions and are not guaranteed to be backward compatible. However UDA version 2.0 is specified as a superset of UDA 1.1 and is thus backwards compatible with UDA 1.x versions. Therefore UDA 2.0 control points shall maintain interoperability with UDA 1.x devices. UDA 1.x control points can work with UDA 2.0 devices, but can't access the additional functionality specified in UDA 2.0. Version information is communicated in discovery and description messages. Discovery messages include the version of UPnP networking that the devices and control points support (in the SERVER and USER-AGENT header fields); the version of device and service types supported is also included in relevant discovery messages. Additionally, description documents also include the same information. SERVER and USER-AGENT header fields are also used in control and eventing to communicate which version of UPnP networking the devices and control points support. This clause explains the format of version information in discovery messages and specific requirements on discovery messages to maintain compatibility with advances in minor versions.

The remainder of this clause explains the UPnP discovery protocol known as SSDP (Simple Service Discovery Protocol) in detail, enumerating how devices advertise and revoke their advertisements as well as how control points search and devices respond.

1.1 SSDP message format

SSDP uses part of the header field format of HTTP 1.1 as defined in RFC 2616. However, it is NOT based on full HTTP 1.1 as it uses UDP instead of TCP, and it has its own processing rules. This subclause defines the generic format of a SSDP message.

All SSDP messages shall be formatted according to RFC 2616 clause 4.1 “generic message”. SSDP messages shall have a start-line and a list of message header fields. SSDP messages should not have a message body. If a SSDP message is received with a message body, the message body is allowed to be ignored.

1.1.1 SSDP Start-line

Each SSDP message shall have exactly one start-line. See clause 1.2, “Advertisement” and clause 1.3, “Search” below for the definition of all possible SSDP messages. The start-line shall be formatted either as defined in RFC 2616 clause 5.1 or clause 6.1. Furthermore, the start-line shall be one of the following three:

```
NOTIFY * HTTP/1.1\r\n
M-SEARCH * HTTP/1.1\r\n
HTTP/1.1 200 OK\r\n
```

As a clarification, while the start-line shall include “HTTP/1.1”, this does not signal that SSDP is fully based on HTTP 1.1; this start-line element is included for backward compatibility reasons only.

1.1.2 SSDP message header fields

The message header fields in a SSDP message shall be formatted according to RFC 2616 clause 4.2. This specifies that each message header field consist of a case-insensitive field name followed by a colon (":"), followed by the case-sensitive field value. SSDP restricts allowed field values.

Example SSDP header:

```
HOST: 239.255.255.250:1900
```

1.1.3 SSDP header field extensions

UPnP working committees and UPnP vendors are allowed to extend SSDP messages with additional SSDP header fields. Additional message header fields can also be defined by the UPnP Forum Technical committee (e.g. clause 1.2, "Advertisement" defines BOOTID.UPNP.ORG, CONFIGID.UPNP.ORG, NEXTBOOTID.UPNP.ORG, and SEARCHPORT.UPNP.ORG header fields). To prevent name-clashes of header field definitions (two parties accidentally define the same header field name with different semantics), vendor-defined header field names shall have the following format:

field-name = token "." domain-name

where the domain-name shall be Vendor Domain Name, and in addition shall satisfy the token format as defined in RFC 2616, clause 2.2.

Example vendor-defined SSDP header fields:

myheader.philips.com: "some value"
myheader.sony.com: "other value"

1.1.4 UUID format and recommended generation algorithms

UPnP 2.0 devices shall format UUIDs according to the format specified below. However, UPnP 2.0 control points shall also be able to accept UUIDs that have not been formatted according to the rules specified below, as formatting rules are not specified in UPnP 1.0 other than the requirement that a UUID is a string.

UUIDs are 128 bit numbers that shall be formatted as specified by the following grammar (taken from [1]):

```
UUID = 4 * <hexOctet> "-" 2 * <hexOctet> "-" 2 * <hexOctet> "-" 2 * <hexOctet> "-" 6 * <hexOctet>
hexOctet = <hexDigit> <hexDigit>
hexDigit = "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"|"a"|"b"|"c"|"d"|"e"|"f"|"A"|"B"|"C"|"D"|"E"|"F"
```

The following is an example of the string representation of a UUID:

"2fac1234-31f8-11b4-a222-08002b34c003"

UUIDs are allowed to be generated using *any* suitable generation algorithm¹ that satisfies the following requirements:

- It is very unlikely to duplicate a UUID generated from some other resource.
- It maps down to a 128-bit number.
- UUIDs shall remain fixed over time.

The following UUID generation algorithm is recommended:

Time & MAC-based algorithm as specified in [1], where the UUID is generated once and stored in non-volatile memory if available.

1.1.5 SSDP processing rules

When an SSDP message is received that is not formatted according to clause 1.1, "SSDP message format" (the clauses above), receivers should silently discard the message. Receivers are allowed to try to parse such SSDP messages to try to interoperate.

When parsing SSDP header fields, receivers shall parse all required SSDP-defined header fields (see clause 1.2, "Advertisement" and clause 1.3, "Search" below) and are allowed to skip all other header fields. Receivers shall be able to skip header fields they do not understand.

¹ The UUID generation algorithm specified in [1] is RECOMMENDED, but is not MANDATORY, other UUID generation algorithms may be used instead, as long as they satisfy the three requirements.

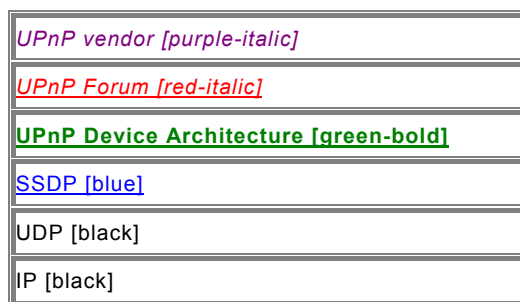
1.2 Advertisement

When a device is added to the network, the device advertises its services to control points. It does this by multicasting discovery messages to a standard address and port (239.255.250:1900). Control points listen to this port to detect when new capabilities are available on the network. To advertise the full extent of its capabilities, a device shall multicast a number of discovery messages corresponding to each of its root devices, embedded devices and services. Each message contains information specific to the embedded device (or service) as well as information about its enclosing device. Messages shall include duration until the advertisements expire; if the device remains available, the advertisements shall be re-sent (with new duration). If the device becomes unavailable, the device should explicitly cancel its advertisements, but if the device is unable to do this, the advertisements will expire on their own. If a multi-homed device becomes unavailable on some, but not all, of its UPnP-enabled interfaces, the device should explicitly cancel its advertisements on the affected UPnP-enabled interfaces (but NOT on the unaffected UPnP-enabled interfaces), but if the device is unable to do this, the advertisements on those interfaces or IP addresses will expire on their own. In addition, messages include the following header fields defined in this document: BOOTID.UPNP.ORG, NEXTBOOTID.UPNP.ORG, CONFIGID.UPNP.ORG, SEARCHPORT.UPNP.ORG. The field value of the BOOTID.UPNP.ORG header field shall be increased each time a device (re)joins the network and sends an initial announce (a “reboot” in UPnP terms), or adds a UPnP-enabled interface. Unless the device explicitly announces a change in the BOOTID.UPNP.ORG field value using an SSDP message, as long as the device remains continuously available in the network, the same BOOTID.UPNP.ORG field value shall be used in all repeat announcements, search responses, update messages and eventually bye-bye messages. Control points can parse this header field to detect whether the device has potentially lost its state (event subscriptions will have been lost, DCP specific state may have been changed) due to a “reboot”. Since a device cannot change IP addresses without changing the BOOTID.UPNP.ORG field value, the BOOTID.UPNP.ORG field value can also be used to distinguish multi-homed devices (in this case, a control point will see SSDP messages from different IP addresses with the same UUID, BOOTID.UPNP.ORG field value) from devices that changed IP addresses (in this case, the BOOTID.UPNP.ORG field value will be different). The field value of the NEXTBOOTID.UPNP.ORG header field indicates the field value of the BOOTID.UPNP.ORG header field that a multi-homed device intends to use in future announcements after adding a new UPnP-enabled interface. The field value of the CONFIGID.UPNP.ORG header field identifies the current set of device and service descriptions; control points can parse this header field to detect whether they need to send new *description* query messages. The field value of the SEARCHPORT.UPNP.ORG header field identifies the port at which the device listens to unicast M-SEARCH messages; control points can parse this header field to know to which port unicast M-SEARCH messages shall be sent. These header fields are explained in detail below.

1.2.1 Advertisement protocols and standards

To send (and receive) advertisements, devices (and control points) use the following subset of the overall UPnP protocol stack. (The overall UPnP protocol stack is listed at the beginning of this document.)

Figure 1-2: — Advertisement protocol stack



At the highest layer, discovery messages contain vendor-specific information, e.g., URL for the device description and device identifier. Moving down the stack, vendor content is supplemented by information from a UPnP Forum working committee, e.g., device type.

Messages from the layers above are hosted in UPnP-specific protocols, defined in this document. In turn, the SSDP messages are delivered via UDP over IP. For reference, colors in [square brackets] above indicate which protocol defines specific header fields and field values in discovery messages listed below.

1.2.2 Device available - NOTIFY with **ssdp:alive**

When a device is added to the network, it shall multicast discovery messages to advertise its root device, any embedded devices, and any services. Each discovery message shall contain four major components:

- A notification type (e.g., device type), sent in an **NT** (Notification Type) header field.
- A composite identifier for the advertisement, sent in a **USN** (Unique Service Name) header field.
- A URL for more information about the device (or enclosing device in the case of a service), sent in a **LOCATION** header field.
- A duration for which the advertisement is valid, sent in a **CACHE-CONTROL** header field.

To advertise its capabilities, a device multicasts a number of discovery messages. Specifically, a root device shall multicast:

- Three discovery messages for the root device.

Table 1-1 — Root device discovery messages

	NT	USN ^a
1	<u>upnp:rootdevice</u>	uuid: <i>device-UUID</i> :: <u>upnp:rootdevice</u>
2	uuid: <i>device-UUID</i> ^b	uuid: <i>device-UUID</i> (for root device UUID)
3	urn: <u>schemas-upnp-org:device:deviceType:ver</u> or urn: <i>domain-name</i> : <u>device:deviceType:ver</u>	uuid: <i>device-UUID</i> ::urn: <u>schemas-upnp-org:device:deviceType:ver</u> (of root device) or uuid: <i>device-UUID</i> ::urn: <i>domain-name</i> : <u>device:deviceType:ver</u>
^a Note that the prefix of the USN header field (before the double colon) shall match the value of the UDN element in the device description. (Clause 2, "Description" explains the UDN element.)		
^b Note that the field value of this NT header field shall match the value of the UDN element in the device description.		

- Two discovery messages for each embedded device.

Table 1-2 — Embedded device discovery messages

	NT	USN ^a
1	uuid: <i>device-UUID</i> ^b	uuid: <i>device-UUID</i>
2	urn: <u>schemas-upnp-org:device:d</u> <u>eviceType:ver</u> or urn: <i>domain-name</i> : <u>device:deviceType:ver</u>	uuid: <i>device-UUID</i> ::urn: <u>schemas-upnp-org:device:deviceType:ver</u> or uuid: <i>device-UUID</i> ::urn: <i>domain-name</i> : <u>device:deviceType:ver</u>
^a Note that the prefix of the USN header field (before the double colon) shall match the value of the UDN element in the device description. (Clause 2, "Description" explains the UDN element.)		
^b Note that the field value of this NT header field shall match the value of the UDN element in the device description		

- Once for each service type in each device.

Table 1-3 — Service discovery messages

	NT	USN ^a
--	----	------------------

	NT	USN ^a
1	urn: <u>schemas-upnp-</u> <u>org:service:serviceType:ver</u> or urn:domain-name: <u>service:serviceType:ver</u>	uuid:device-UUID::urn: <u>schemas-upnp-</u> <u>org:service:serviceType:ver</u> or uuid:device-UUID::urn:domain- name: <u>service:serviceType:ver</u>
a	Note that the field value of this NT header field shall match the value of the UDN element in the device description.	

If a root device has d embedded devices and s embedded services but only k distinct service types, this works out to $3+2d+k$ requests. If a particular device or embedded device contains multiple instances of a particular service type, it is only necessary to advertise the service type once (rather than once for each instance). Note that if two embedded devices contain a service of the same service type, these services shall still be separately announced. This advertises the full extent of the device's capabilities to interested control points. These messages shall be sent out as a series with roughly comparable expiration times; order is unimportant, but refreshing or canceling individual messages is PROHIBITED.

Updated UPnP device and service types are required to be fully backward compatible with previous versions of the same type. Devices shall advertise the highest supported version of each supported type. For example, if a device supports version 2 of the "Audio" service, it would advertise only version 2, even though it also supports version 1. It shall NOT advertise additional supported versions. Control points that support a given version of a device or service are able to also interact with higher versions because of this backward compatibility requirement, but only using the functionality that was defined in the lower version. For example, if a control point supports only version "1" of the "Audio" service, and a device advertises that it supports version "2" of the "Audio" service, the control point shall recognize the device and be able to use it.

Choosing an appropriate duration for advertisements is a balance between minimizing network traffic and maximizing freshness of device status. Relatively short durations close to the minimum of 1800 seconds will ensure that control points have current device status at the expense of additional network traffic; longer durations, say on the order of a day, compromise freshness of device status but can significantly reduce network traffic. Generally, device vendors should choose a value that corresponds to expected device usage: short durations for devices that are expected to be part of the network for short periods of time, and significantly longer durations for devices expected to be long-term members of the network. Devices that frequently connect to and leave the network (such as mobile wireless devices) should use a shorter duration so that control points have a more accurate view of their availability. Advertisements in a set (both initial and subsequent) should have comparable durations. Advertisements in the initial set should be sent as quickly as possible. Subsequent refreshments of the advertisements are allowed to be spread over time rather than being sent as a group.

Spreading refreshments of advertisements over time rather than being sent as a group improves reliability in case there are network glitches: without increasing the total network load it increases the frequency of sending announcements from devices to control points. The two figures below show the announcement behavior without spreading and with spreading the messages over the entire interval. The figures show a timeline from the moment a device joins the network, sends its initial announcements (represented by vertical lines), and subsequently periodically sends repeat announcements. In the second figure, these repeat announcements are spread over the entire period rather than sent as a bunch.

Figure 1-3: — Initial and repeat announcements, no announcement spreading

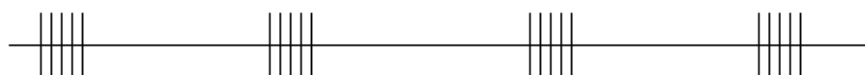


Figure 1-4: — Initial and repeat announcements, message spreading of repeat announcements



Devices should wait a random interval (e.g. between 0 and 100 milliseconds) before sending an initial set of advertisements in order to reduce the likelihood of network storms; this random interval should also be applied on occasions where the device obtains a new IP address or a new UPnP-enabled interface is installed.

Due to the unreliable nature of UDP, devices should send the entire set of discovery messages more than once with some delay between sets e.g. a few hundred milliseconds. To avoid network congestion discovery messages should not be sent more than three times. In addition, the device shall re-send its advertisements periodically prior to expiration of the duration specified in the **CACHE-CONTROL** header field; it is Recommended that such refreshing of advertisements be done at a randomly-distributed interval of less than one-half of the advertisement expiration time, so as to provide the opportunity for recovery from lost advertisements before the advertisement expires, and to distribute over time the advertisement refreshment of multiple devices on the network in order to avoid spikes in network traffic. Note that UDP packets are also bounded in length (perhaps as small as 512 Bytes in some implementations); each discovery message shall fit entirely in a single UDP packet. There is no guarantee that the above $3+2d+k$ messages will arrive in a particular order.

A multi-homed device shall perform the above announcement procedures on each of its UPnP-enabled interfaces. Advertisements sent on multiple UPnP-enabled interfaces shall contain the same field values except for the HOST, CACHE-CONTROL and LOCATION header fields. The HOST field value of an advertisement shall be the standard multicast address specified for the protocol (IPv4 or IPv6) used on the interface. The URL specified by the LOCATION header field shall be reachable on the interface on which the advertisement is sent. Finally, advertisements sent on different interfaces are allowed to have different CACHE-CONTROL field values and are allowed to be sent with different frequencies.

When a device is added to the network, it shall send a multicast message with method **NOTIFY** and **ssdp:alive** in the **NTS** header field in the following format. Values in *italics* are placeholders for actual values.

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
CACHE-CONTROL: max-age = seconds until advertisement expires
LOCATION: URL for UPnP description for root device
NT: notification type
NTS: ssdp:alive
SERVER: OS/version UPnP/2.0 product/version
USN: composite identifier for the advertisement
BOOTID.UPNP.ORG: number increased each time device sends an initial announce or an update message
CONFIGID.UPNP.ORG: number used for caching description information
SEARCHPORT.UPNP.ORG: number identifies port on which device responds to unicast M-SEARCH
```

Note: No body is sent for messages with method **NOTIFY**, but note that the message shall have a blank line following the last header field.

The TTL for the IP packet should default to 2 and should be configurable.

Listed below are details for the request line and header fields appearing in the listing above. Field names are not case sensitive. All field values are case sensitive except where noted.

Request line

Shall be "NOTIFY * HTTP/1.1"

NOTIFY

Method for sending notifications and events.

*

Message applies generally and not to a specific resource. shall be *.

HTTP/1.1

HTTP version.

Header fields

HOST

Required. Field value contains multicast address and port reserved for SSDP by Internet Assigned Numbers Authority (IANA). shall be [239.255.255.250:1900](#). If the port number (":1900") is omitted, the receiver shall assume the default SSDP port number of 1900.

CACHE-CONTROL

Required. Field value shall have the max-age directive ("max-age=") followed by an integer that specifies the number of seconds the advertisement is valid. After this duration, control points should assume the device (or service) is no longer available; as long as a control point has received at least one advertisement that is still valid from a root device, any of its embedded devices or any of its services, then the control point can assume that all are available. The number of seconds should be greater than or equal to 1800 seconds (30 minutes), although exceptions are defined in the text above. Specified by UPnP vendor. Other directives shall NOT be sent and shall be ignored when received.

LOCATION

Required. Field value contains a URL to the UPnP description of the root device. Normally the host portion contains a literal IP address rather than a domain name in unmanaged networks. Specified by UPnP vendor. Single absolute URL (see RFC 3986).

NT

Required. Field value contains Notification Type. shall be one of the following. (See Table 1-1, "Root device discovery messages", Table 1-2, "Embedded device discovery messages", and Table 1-3, "Service discovery messages" above.) Single URI.

[upnp:rootdevice](#)

Sent once for root device.

[uuid:device-UUID](#)

Sent once for each device, root or embedded, where [device-UUID](#) is specified by the UPnP vendor. See clause 1.1.4, "UUID format and recommended generation algorithms" for the MANDATORY UUID format.

[urn:schemas-upnp-org:device:deviceType:ver](#)

Sent once for each device, root or embedded, where [deviceType](#) and [ver](#) are defined by UPnP Forum working committee, and [ver](#) specifies the version of the device type.

[urn:schemas-upnp-org:service:serviceType:ver](#)

Sent once for each service where [serviceType](#) and [ver](#) are defined by UPnP Forum working committee and [ver](#) specifies the version of the service type.

[urn:domain-name:device:deviceType:ver](#)

Sent once for each device, root or embedded, where [domain-name](#) is a Vendor Domain Name, [deviceType](#) and [ver](#) are defined by the UPnP vendor, and [ver](#) specifies the version of the device type. Period characters in the Vendor Domain Name shall be replaced with hyphens in accordance with RFC 2141.

[urn:domain-name:service:serviceType:ver](#)

Sent once for each service where [domain-name](#) is a Vendor Domain Name, [serviceType](#) and [ver](#) are defined by UPnP vendor, and [ver](#) specifies the version of the service type. Period characters in the Vendor Domain Name shall be replaced with hyphens in accordance with RFC 2141.

NTS

Required. Field value contains Notification Sub Type. shall be [ssdp:alive](#). Single URI.

SERVER

Required. Specified by UPnP vendor. String. Field value shall begin with the following “product tokens” (defined by HTTP/1.1). The first product token identifies the operating system in the form *OS name/OS version*, the second token represents the UPnP version and shall be *UPnP/2.0*, and the third token identifies the product using the form *product name/product version*. For example, “SERVER: *unix/5.1 UPnP/2.0 MyProduct/1.0*”.

USN

Required. Field value contains Unique Service Name. Identifies a unique instance of a device or service. shall be one of the following. (See Table 1-1, “Root device discovery messages”, Table 1-2, “Embedded device discovery messages”, and Table 1-3, “Service discovery messages” above.) The prefix (before the double colon) shall match the value of the UDN element in the device description. (Clause 2, “Description” explains the UDN element.) Single URI.

uuid:*device-UUID*::*upnp:rootdevice*

Sent once for root device where *device-UUID* is specified by UPnP vendor. See clause 1.1.4, “UUID format and recommended generation algorithms” for the MANDATORY UUID format.

uuid:*device-UUID*

Sent once for every device, root or embedded, where *device-UUID* is specified by the UPnP vendor. See clause 1.1.4, “UUID format and recommended generation algorithms” for the MANDATORY UUID format.

uuid:*device-UUID*::urn:*schemas-upnp-org:device:deviceType:ver*

Sent once for every device, root or embedded, where *device-UUID* is specified by the UPnP vendor, *deviceType* and *ver* are defined by UPnP Forum working committee and *ver* specifies version of the device type. See clause 1.1.4, “UUID format and recommended generation algorithms” for the MANDATORY UUID format.

uuid:*device-UUID*::urn:*schemas-upnp-org:service:serviceType:ver*

Sent once for every service where *device-UUID* is specified by the UPnP vendor, *serviceType* and *ver* are defined by UPnP Forum working committee and *ver* specifies version of the device type. See clause 1.1.4, “UUID format and recommended generation algorithms” for the MANDATORY UUID format.

uuid:*device-UUID*::urn:*domain-name:device:deviceType:ver*

Sent once for every device, root or embedded, where *device-UUID*, *domain-name* (a Vendor Domain Name), *deviceType* and *ver* are defined by the UPnP vendor and *ver* specifies the version of the device type. See clause 1.1.4, “UUID format and recommended generation algorithms” for the MANDATORY UUID format. Period characters in the Vendor Domain Name shall be replaced by hyphens in accordance with RFC 2141.

uuid:*device-UUID*::urn:*domain-name:service:serviceType:ver*

Sent once for every service where *device-UUID*, *domain-name* (a Vendor Domain Name), *serviceType* and *ver* are defined by the UPnP vendor and *ver* specifies the version of the service type. See clause 1.1.4, “UUID format and recommended generation algorithms” for the MANDATORY UUID format. Period characters in the Vendor Domain Name shall be replaced by hyphens in accordance with RFC 2141.

BOOTID.UPNP.ORG

Required. The BOOTID.UPNP.ORG header field represents the boot instance of the device expressed according to a monotonically increasing value. Its field value shall be a non-negative 31-bit integer; ASCII encoded, decimal, without leading zeros (leading zeroes, if present, shall be ignored by the recipient) that shall be increased on each initial announce of the UPnP device or shall be the same as the field value of the NEXTBOOTID.UPNP.ORG header field in the last sent SSDP update message. Its field value shall remain the same on all periodically repeated announcements. A convenient mechanism is to set this field value to the time that the device sends its initial announcement, expressed as seconds elapsed since midnight January 1, 1970; for devices that have a notion of time, this will not require any additional state to remember or be “flushed”. However, it is perfectly acceptable to use a simple boot counter that is incremented on every initial announcement as a field value of this header field. As such, control points shall NOT view this header field as a timestamp. The BOOTID.UPNP.ORG header field shall be included in *all* announcements of a root device, its embedded devices and its services. Unless the device explicitly updates its value by sending an SSDP update message, as long as the device remains available in the network, the same BOOTID.UPNP.ORG field value shall be used in *all* announcements, search responses, update messages and eventually bye-bye messages.

Control points can use this header field to detect the case when a device leaves and rejoins the network (“reboots” in UPnP terms). It can be used by control points for a number of purposes such as re-establishing desired event subscriptions, checking for changes to the device state that were not evented since the device was off-line.

CONFIGID.UPNP.ORG

Required. The CONFIGID.UPNP.ORG field value shall be a non-negative, 31-bit integer, ASCII encoded, decimal, without leading zeros (leading zeroes, if present, shall be ignored by the recipient) that shall represent the configuration number of a root device. UPnP 2.0 devices are allowed to be freely assign configid numbers from 0 to 16777215 ($2^{24}-1$). Higher numbers are reserved for future use, and can be assigned by the Technical Committee. The **configuration** of a root device consists of the following information: the DDD of the root device and all its embedded devices, and the SCPDs of all the contained services. If any part of the configuration changes, the CONFIGID.UPNP.ORG field value shall be changed. The CONFIGID.UPNP.ORG header field shall be included in all announcements of a root device, its embedded devices and its services. The configuration number that is present in a CONFIGID.UPNP.ORG field value shall satisfy the following rule:

- if a device sends out two messages with a CONFIGID.UPNP.ORG header field with the same field value K, the configuration shall be the same at the moments that these messages were sent.

Whenever a control point receives a CONFIGID.UPNP.ORG header field with a field value K, and subsequently downloads the configuration information, this configuration information is associated with K. As an additional safeguard, the device shall include a `configId` attribute with value K in the returned description (see clause 2, "Description"). The following caching rules for control points supersede the caching rules that are defined in UPnP 1.0:

- Control points are allowed to ignore the CONFIGID.UPNP.ORG header field and use the caching rules that are based on advertisement expirations as defined in Clause 2, Description: as long as at least one of the discovery advertisements from a root device, its embedded devices and its services have not expired, a control point is allowed to assume that the root device and all its embedded devices and all its services are available. The device and service descriptions are allowed to be retrieved at any point since the device and service descriptions are static as long as the device and its services are available.
- If no configuration number is included in a received SSDP message, control points should cache based on advertisement expirations as defined in Clause 2 Description.
- If a CONFIGID.UPNP.ORG header field with field value K is included in a received SSDP message, and a control point has already cached information associated with field value K, the control point is allowed to use this cached information as the current configuration of the device. Otherwise, a control point should assume it has not cached the current configuration of the device and needs to send new description query messages.

The CONFIGID.UPNP.ORG header field reduces peak loads on UPnP devices during startup and during network hiccups. Only if a control point receives an announcement of an unknown configuration is downloading required.

SEARCHPORT.UPNP.ORG

Allowed. If a device does not send the SEARCHPORT.UPNP.ORG header field, it shall respond to unicast M-SEARCH messages on port 1900. Only if port 1900 is unavailable it is allowed for a device select a different port to respond to unicast M-SEARCH messages. If a device sends the SEARCHPORT.UPNP.ORG header field, its field value shall be an ASCII encoded integer, decimal, without leading zeros (leading zeroes, if present, shall be ignored by the recipient), in the range 49152-65535 (RFC 4340). The device shall respond to unicast M-SEARCH messages that are sent to the advertised port.

SECURELOCATION.UPNP.ORG

Allowed. Required when Device Protection is implemented.

The SECURELOCATION.UPNP.ORG header shall provide a base URL with "https:" for the scheme component and indicate the correct "port" subcomponent in the "authority" component for a TLS connection. Because the scheme and authority components are not included in relative URLs, these components are obtained from the base URL provided by either LOCATION or SECURELOCATION.UPNP.ORG. See for more information Ref DEVICEPROTECTION.

Note: No responses are sent for messages with method NOTIFY.

1.2.3 Device unavailable -- NOTIFY with ssdp:byebye

When a device and its services are going to be removed from the network, the device should multicast an **ssdp:byebye** message corresponding to *each* of the **ssdp:alive** messages it multicastrd that have not already expired. If the device is removed abruptly from the network, it might not be possible to multicast a message. As a fallback, discovery messages shall include an expiration value in a CACHE-CONTROL field value (as explained above); if not re-advertised, the discovery message eventually expires on its own.

(Note: when a control point is about to be removed from the network, no discovery-related action is required.)

When a device is about to be removed from the network, it should explicitly revoke its discovery messages by sending one multicast message for *each* **ssdp:alive** message it sent.

Each multicast message shall have method **NOTIFY** and **ssdp:byebye** in the **NTS** header field in the following format. Values in *italics* are placeholders for actual values.

When a multi-homed device is about to be removed from the network on one or more of its UPnP-enabled interfaces, it should explicitly revoke its discovery messages by sending one multicast message for *each* **ssdp:alive** message it has previously sent on those interfaces and IP addresses. It shall NOT send such multicast messages to any of the UPnP-enabled interfaces that remain available.

When **ssdp:byebye** messages are sent on multiple UPnP-enabled interfaces, the messages shall contain identical field values except for the HOST field value. The HOST field value of an advertisement shall be the standard multicast address specified for the protocol (IPv4 or IPv6) used on the interface.

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
NT: notification type
NTS: ssdp:byebye
USN: composite identifier for the advertisement
BOOTID.UPNP.ORG: number increased each time device sends an initial announce or an update message
CONFIGID.UPNP.ORG: number used for caching description information
```

Note: No body is present for messages with method NOTIFY, but note that the message shall have a blank line following the last header field.

The TTL for the IP packet should default to 2 and should be configurable.

Listed below are details for the request line and header fields appearing in the listing above. Field names are not case sensitive. All field values are case sensitive except where noted.

Request line

Shall be "NOTIFY * HTTP/1.1"

NOTIFY

Method for sending notifications and events.

*

Message applies generally and not to a specific resource. shall be *.

HTTP/1.1

HTTP version.

Header fields

HOST

Required. Field value contains multicast address and port reserved for SSDP by Internet Assigned Numbers Authority (IANA). shall be [239.255.255.250:1900](#). If the port number (":1900") is omitted, the receiver shall assume the default SSDP port number of 1900.

NT

Required. Field value contains Notification Type. (See list of required field values for the NT header field in NOTIFY with ssdp:alive above.) Single URI.

NTS

Required. Field value contains Notification Sub Type. shall be [ssdp:byebye](#). Single URI.

USN

Required. Field value contains Unique Service Name. (See list of required field values for the USN header field in NOTIFY with ssdp:alive above.) Single URI.

BOOTID.UPNP.ORG

Required. As defined in clause 1.2, and 1.2.2.

CONFIGID.UPNP.ORG

Required. As defined in clause 1.2, and 1.2.2.

Note: No responses are sent for messages with method NOTIFY.

If a control point has received *at least one* ssdp:byebye message of a root device, any of its embedded devices or any of its services then the control point can assume that all are no longer available. As a fallback, if a control point fails to receive notification that a root device, its embedded devices and its services are unavailable, the original discovery messages will eventually expire yielding the same effect. Only when *all* original advertisements of a root device, its embedded devices and its services have expired can a control point assume that they are no longer available.

If a multi-homed control point has received *at least one* ssdp:byebye message of a root device, any of its embedded devices or any of its services on one of its UPnP-enabled interfaces then the control point can assume that all are no longer available on that UPnP-enabled interface. However, the control point shall NOT assume that the device is also no longer available on all of its other UPnP-enabled interfaces. As a fallback, if a control point fails to receive notification that a root device, its embedded devices and its services are unavailable on a particular UPnP-enabled interface, the original discovery messages will eventually expire yielding the same effect. Only when *all* original advertisements of a root device, its embedded devices and its services received on a UPnP-enabled interface have expired can a control point assume that they are no longer available on that interface or IP address.

1.2.4 Device Update – NOTIFY with ssdp:update

When a new UPnP-enabled interface is added to a multi-homed device, the device shall increase its BOOTID.UPNP.ORG field value, multicast an **ssdp:update** message for each of the root devices, embedded devices and embedded services to all of the existing UPnP-enabled interfaces to announce a change in the BOOTID.UPNP.ORG field value, and re-advertise itself on all (existing and new) UPnP-enabled interfaces with the new BOOTID.UPNP.ORG field value. Similarly, if a multi-homed device loses connectivity on a UPnP-enabled interface and regains connectivity, or if the IP address on one of the UPnP-enabled interfaces changes, the device shall increase the BOOTID.UPNP.ORG field value, multicast an **ssdp:update** message for each of the root devices, embedded devices and embedded services to all the unaffected UPnP-enabled interfaces to announce a change in the BOOTID.UPNP.ORG field value, and re-advertise itself on all (affected and unaffected) UPnP-enabled interfaces with the new BOOTID.UPNP.ORG field value. In all cases, the **ssdp:update** message for the root devices shall be sent as soon as possible. Other **ssdp:update** messages should be spread over time. However, all **ssdp:update** messages shall be sent before any announcement messages with the new BOOTID.UPNP.ORG field value can be sent.

When **ssdp:update** messages are sent on multiple UPnP-enabled interfaces, the messages shall contain identical field values except for the HOST and LOCATION field values. The HOST field value of an advertisement shall be the standard multicast address specified for the protocol (IPv4 or IPv6) used on the interface. The URL specified in the LOCATION field value shall be reachable on the interface on which the advertisement is sent.

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
LOCATION: URL for UPnP description for root device
NT: notification type
NTS: ssdp:update
USN: composite identifier for the advertisement
BOOTID.UPNP.ORG: BOOTID value that the device has used in its previous announcements
CONFIGID.UPNP.ORG: number used for caching description information
NEXTBOOTID.UPNP.ORG: new BOOTID value that the device will use in subsequent announcements
SEARCHPORT.UPNP.ORG: number identifies port on which device responds to unicast M-SEARCH
```

Note: No body is present for messages with method NOTIFY, but note that the message shall have a blank line following the last header field.

The TTL for the IP packet should default to 2 and should be configurable.

© ISO/IEC 2017 – All rights reserved

Listed below are details for the request line and header fields appearing in the listing above. Field names are not case sensitive. All field values are case sensitive except where noted.

Request line

Shall be "NOTIFY * HTTP/1.1"

NOTIFY

Method for sending notifications and events.

*

Message applies generally and not to a specific resource. Shall be *.

HTTP/1.1

HTTP version.

Header fields

HOST

Required. Field value contains multicast address and port reserved for SSDP by Internet Assigned Numbers Authority (IANA). Shall be [239.255.255.250:1900](#). If the port number (":1900") is omitted, the receiver shall assume the default SSDP port number of 1900.

LOCATION

Required. Field value shall be the same as the LOCATION field value that has been sent in previous SSDP messages. Single absolute URL (see RFC 3986).

NT

Required. Field value contains Notification Type. (See list of required field values for the NT header field in NOTIFY with ssdp:alive above.) Single URI.

NTS

Required. Field value contains Notification Sub Type. Shall be [ssdp:update](#). Single URI.

USN

Required. Field value contains Unique Service Name. (See list of required field values for the USN header field in NOTIFY with ssdp:alive above.) Single URI.

BOOTID.UPNP.ORG

Required. As defined in clause 1.2, and 1.2.2, Field value shall be the same as the BOOTID.UPNP.ORG field value that has been sent in previous SSDP messages.

CONFIGID.UPNP.ORG

Required. As defined in clause 1.2, and 1.2.2.

NEXTBOOTID.UPNP.ORG

Required. Field value contains the new BOOTID.UPNP.ORG field value that the device intends to use in the subsequent device and service announcement messages. Its field value shall be a non-negative 31-bit integer; ASCII encoded, decimal, without leading zeros (leading zeroes, if present, shall be ignored by the recipient) and shall be greater than the field value of the BOOTID.UPNP.ORG header field.

SEARCHPORT.UPNP.ORG

Allowed. As defined in clause 1.2, and 1.2.2.

SECURELOCATION.UPNP.ORG

Allowed. As defined in section 1.2.2.

Note: No responses are sent for messages with method NOTIFY.

If a control point with a single UPnP-enabled interface receives an **ssdp:update** message, the NEXTBOOTID.UPNP.ORG field value replaces the BOOTID.UPNP.ORG field value that the control point has previously recorded for the device. It can expect future announcements, search responses, update messages and eventually bye-bye messages from the device to contain the "new" BOOTID.UPNP.ORG field value (that is: the field value of the NEXTBOOTID.UPNP.ORG header field in the received **ssdp:update** message). The field value in the NEXTBOOTID.UPNP.ORG header field shall be recorded as the current

BOOTID.UPNP.ORG field value of the device which is to be expected on all subsequent SSDP messages.

If a multi-homed control point receives an **ssdp:update** message on its UPnP-enabled interface(s), and the message arrives on the interface(s) that it uses for UPnP communications with the device (such as event subscriptions), it can assume that the device has remained continuously available (including all device state), and that the NEXTBOOTID.UPNP.ORG field value replaces the BOOTID.UPNP.ORG field value that the control point has previously recorded for the device. It can expect future announcements, search responses, update messages and eventually bye-bye messages from the device to contain the “new” BOOTID.UPNP.ORG field value (that is: the field value of the NEXTBOOTID.UPNP.ORG header field in the received **ssdp:update** message). The field value in the NEXTBOOTID.UPNP.ORG header field shall be recorded as the current BOOTID.UPNP.ORG field value of the device which is to be expected on all subsequent SSDP messages.

If a control point receives an SSDP message with a BOOTID.UPNP.ORG field value different (either higher or lower) from the value that the control point has previously recorded for the device, it can assume that the device has become temporarily unavailable on that interface and has become available again, and any stored state information about the device has become invalid. It shall treat the device as a newly discovered device.

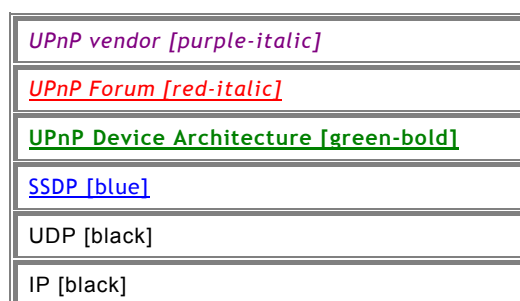
1.3 Search

When a control point is added to the network, the UPnP discovery protocol allows that control point to search for devices of interest on the network. It does this by multicasting on the reserved address and port (239.255.255.250:1900) a search message with a pattern, or target, equal to a type or identifier for a device or service. Responses from devices contain discovery messages essentially identical to those advertised by newly connected devices; the former are unicast while the latter are multicast. Control points can also send a unicast search message to a known IP address and port 1900 or the port indicated by SEARCHPORT.UPNP.ORG, to verify the existence of UPnP device(s) and service(s) at the IP address. For example, a unicast search may be used to quickly check whether a known UPnP device or service is still available on the network. Multi-homed control points are allowed to choose to send discovery messages on any, some or all of its UPnP-enabled interfaces.

1.3.1 Search protocols and standards

To search for devices (and be discovered by control points), control points (and devices) use the following subset of the overall UPnP protocol stack. (The overall UPnP protocol stack is listed at the beginning of this document.)

Figure 1-5: — Search protocol stack



At the highest layer, search messages contain vendor-specific information, e.g., the control point, device, and service identifiers. Moving down the stack, vendor content is supplemented by information from a UPnP Forum working committee, e.g., device or service types. Messages from the layers above are hosted in UPnP-specific protocols, defined in this document. In turn, search requests are delivered via multicast and unicast SSDP messages defined in this document. Search responses are delivered via a unicast SSDP messages defined in this document. Both kinds of messages are delivered via UDP over IP. For

reference, colors in [square brackets] above indicate which protocol defines specific header fields and field values in discovery messages listed below.

1.3.2 Search request with M-SEARCH

When a control point desires to search the network for devices, it shall send a multicast request with method **M-SEARCH** in the following format. Control points that know the address of a specific device are allowed to also use a similar format to send unicast requests with method M-SEARCH.

For multicast M-SEARCH, the message format is defined below. Values in *italics* are placeholders for actual values.

```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
MX: seconds to delay response
ST: search target
USER-AGENT: OS/version UPnP/2.0 product/version
CPFN.UPNP.ORG: friendly name of the control point
CPUUID.UPNP.ORG: uuid of the control point
```

Note: No body is present in requests with method **M-SEARCH**, but note that the message shall have a blank line following the last header field.

Note: The TTL for the IP packet should default to 2 and should be configurable.

Listed below are details for the request line and header fields appearing in the listing above. Field names are not case sensitive. All field values are case sensitive except where noted.

Request line

Shall be "M-SEARCH * HTTP/1.1"

M-SEARCH

Method for search requests.

*

Request applies generally and not to a specific resource. shall be *.

HTTP/1.1

HTTP version.

Header fields

HOST

Required. Field value contains the multicast address and port reserved for SSDP by Internet Assigned Numbers Authority (IANA). shall be 239.255.255.250:1900.

MAN

Required by HTTP Extension Framework. Unlike the NTS and ST field values, the field value of the MAN header field is enclosed in double quotes; it defines the scope (namespace) of the extension. shall be "ssdp:discover".

MX

Required. Field value contains maximum wait time in seconds. shall be greater than or equal to 1 and should be less than 5 inclusive. Device responses should be delayed a random duration between 0 and this many seconds to balance load for the control point when it processes responses. This value is allowed to be increased if a large number of devices are expected to respond. The MX field value should NOT be increased to accommodate network characteristics such as latency or propagation delay (for more details, see the explanation below). Specified by UPnP vendor. Integer.

ST

Required. Field value contains Search Target. shall be one of the following. (See NT header field in NOTIFY with ssdp:alive above.) Single URI.

ssdp:all

Search for all devices and services.

upnp:rootdevice

Search for root devices only.

uuid:*device-UUID*

Search for a particular device. *device-UUID* specified by UPnP vendor. See clause 1.1.4, "UUID format and recommended generation algorithms" for the MANDATORY UUID format.

urn:schemas-upnp-org:device:*deviceType*:*ver*

Search for any device of this type where *deviceType* and *ver* are defined by the UPnP Forum working committee.

urn:schemas-upnp-org:service:*serviceType*:*ver*

Search for any service of this type where *serviceType* and *ver* are defined by the UPnP Forum working committee.

urn:*domain-name*:device:*deviceType*:*ver*

Search for any device of this type where *domain-name* (a Vendor Domain Name), *deviceType* and *ver* are defined by the UPnP vendor and *ver* specifies the highest supported version of the device type. Period characters in the Vendor Domain Name shall be replaced with hyphens in accordance with RFC 2141.

urn:*domain-name*:service:*serviceType*:*ver*

Search for any service of this type. Where *domain-name* (a Vendor Domain Name), *serviceType* and *ver* are defined by the UPnP vendor and *ver* specifies the highest supported version of the service type. Period characters in the Vendor Domain Name shall be replaced with hyphens in accordance with RFC 2141.

USER-AGENT

Allowed. Specified by UPnP vendor. String. Field value shall begin with the following "product tokens" (defined by HTTP/1.1). The first product token identifies the operating system in the form *OS name/OS version*, the second token represents the UPnP version and shall be UPnP/2.0, and the third token identifies the product using the form *product name/product version*. For example, "USER-AGENT: *unix/5.1* UPnP/2.0 *MyProduct/1.0*".

TCPPOINT.UPNP.ORG

Allowed. A control point can request that a device replies to a TCP port on the control point. When this header is present it identifies the TCP port on which the device can reply to the search. If a control point sends the TCPPOINT.UPNP.ORG header field, its field value shall be an ASCII encoded integer, decimal, without leading zeros (leading zeroes, if present, shall be ignored by the recipient), in the range 49152-65535 (RFC 4340). The device shall respond to unicast M-SEARCH messages similar to sending the response to the originating UDP port except that the notification messages are sent to the advertised TCPPOINT.UPNP.ORG port over TCP instead of UDP.

CPFN.UPNP.ORG

Required. Specifies the friendly name of the control point. The friendly name is vendor specific. When Device Protection is implemented the cpfn.upnp.org shall be the same as the <Name> of Device Protection unless the Device Protection <Alias> is defined, in which case it shall use the <Alias>.

CPUUID.UPNP.ORG

Allowed. uuid of the control point. When the control point is implemented in a UPnP device it is recommended to use the UDN of the co-located UPnP device. When implemented, all specified requirements for uuid usage in devices also apply for control points. See section 1.1.4. Note that when Device Protection is implemented the CPUUID.UPNP.ORG shall be the same as the uuid used in Device Protection.

For unicast M-SEARCH, the message format is defined below. Values in *italics* are placeholders for actual values.

```
M-SEARCH * HTTP/1.1
HOST: hostname:portNumber
MAN: "ssdp:discover"
ST: search target
USER-AGENT: OS/version UPnP/2.0 product/version
```

Note: No body is present in requests with method **M-SEARCH**, but note that the message shall have a blank line following the last header field.

Listed below are details for the request line and header fields appearing in the listing above. Field names are not case sensitive. All field values are case sensitive except where noted.

Request line

Shall be "[M-SEARCH * HTTP/1.1](#)"

M-SEARCH

Method for search requests.

*

Request applies generally and not to a specific resource. Shall be *.

HTTP/1.1

HTTP version.

Header fields

HOST

Required. For unicast requests, the field value shall be the domain name or IP address of the target device and either port 1900 or the SEARCHPORT provided by the target device.

MAN

Required by HTTP Extension Framework. Unlike the NTS and ST field values, the field value of the MAN header field is enclosed in double quotes; it defines the scope (namespace) of the extension. Shall be "[ssdp:discover](#)".

ST

Required. Field value contains Search Target. Shall be one of the following. (See NT header field in NOTIFY with [ssdp:alive](#) above.) Single URI.

[ssdp:all](#)

Search for all devices and services.

[upnp:rootdevice](#)

Search for root devices only.

uuid:*device-UUID*

Search for a particular device. *device-UUID* specified by UPnP vendor. See clause 1.1.4, "UUID format and recommended generation algorithms" for the MANDATORY UUID format.

urn:[schemas-upnp-org:device:deviceType:ver](#)

Search for any device of this type where *deviceType* and *ver* are defined by the UPnP Forum working committee.

urn:[schemas-upnp-org:service:serviceType:ver](#)

Search for any service of this type where *serviceType* and *ver* are defined by the UPnP Forum working committee.

urn:*domain-name*:[device:deviceType:ver](#)

Search for any device of this type where *domain-name* (a Vendor Domain Name), *deviceType* and *ver* are defined by the UPnP vendor and *ver* specifies the highest supported version of the device type. Period characters in the Vendor Domain Name shall be replaced with hyphens in accordance with RFC 2141.

urn:*domain-name*:[service:serviceType:ver](#)

Search for any service of this type where *domain-name* (a Vendor Domain Name), *serviceType* and *ver* are defined by the UPnP vendor and *ver* specifies the highest supported version of the service type. Period characters in the Vendor Domain Name shall be replaced with hyphens in accordance with RFC 2141.

USER-AGENT

Allowed. Specified by UPnP vendor. String. Field value shall begin with the following "product tokens" (defined by HTTP/1.1). The first product token identifies the operating system in the form *OS name/OS version*. The

second token represents the UPnP version and shall be [UPnP/2.0](#), and the third token identifies the product using the form *product name/product version*. For example, "USER-AGENT: *unix/5.1* [UPnP/2.0](#) *MyProduct/1.0*".

Due to the unreliable nature of UDP, control points should send each M-SEARCH message more than once. As a fallback, to guard against the possibility that a device might not receive the M-SEARCH message from a control point, a device should re-send its advertisements periodically (see **CACHE-CONTROL** header field in **NOTIFY** with `ssdp:alive` above).

For a multicast request, the control point should wait at least the amount of time specified in the MX header field for responses to arrive from devices. The random distribution of responses over the MX interval means that a responder is allowed to send a response at MX seconds after receiving the M-SEARCH request. The MX field value is allowed to be adjusted by heuristics at the requester based on, for example, observed number of responders. Network characteristics affecting the propagation of traffic cannot be addressed by increasing the MX field value because of the reason cited above. A requester is allowed to adapt to network characteristics with heuristics based on observed network behavior (the exact heuristics are out of scope). The net effect is that the M-SEARCH request persists at the requester for a period of time exceeding MX such that the characteristics of the network are properly accommodated to minimize lost responses.

When a device receives a unicast M-SEARCH, it should respond within 1 second and it is allowed to respond sooner. The sender of the unicast request should wait at least 1 second for the response.

Updated versions of device and service types are required to be fully backward compatible with previous versions. Devices shall respond to M-SEARCH requests for any supported version. For example, if a device implements "[urn:schemas-upnp-org:service:xyz:2](#)", it shall respond to search requests for both that type and "[urn:schemas-upnp-org:service:xyz:1](#)". The response shall specify the same version as was contained in the search request. If a control point searches for a device or service of a particular version and receives no responses (presumably because no device present on the network supports the specified version), but is willing to operate using a lower version, it is allowed to repeat the search request specifying the lower version.

1.3.3 Search response

To be found by a network search, a device shall send a unicast UDP response to the source IP address and port that sent the request to the multicast address. Devices respond if the ST header field of the M-SEARCH request is "ssdp:all", "upnp:rootdevice", "uuid:" followed by a UUID that exactly matches the one advertised by the device, or if the M-SEARCH request matches a device type or service type supported by the device. Multi-homed devices shall send the search response using the same UPnP-enabled interface on which the search request was received. The URL specified in the LOCATION field value shall specify an address that is reachable on that interface.

Devices responding to a multicast M-SEARCH should wait a random period of time between 0 seconds and the number of seconds specified in the MX field value of the search request before responding, in order to avoid flooding the requesting control point with search responses from multiple devices. If the search request results in the need for a multiple part response from the device, those multiple part responses should be spread at random intervals through the time period from 0 to the number of seconds specified in the [MX](#) header field. Devices are allowed to assume an MX field value less than that specified in the [MX](#) header field. If the [MX](#) header field specifies a field value greater than 5, the device should assume that it contained the value 5 or less. Devices shall not stop responding to other requests while waiting the random delay before sending a response.

For multicast M-SEARCH requests, if the search request does not contain an [MX](#) header field, the device shall silently discard and ignore the search request. If the [MX](#) header field specifies a field value greater than 5, the device should assume that it contained the value 5 or less.

For multicast M-SEARCH requests, if the search request does contain the TCP:PORT.UPNP.ORG header field, the device shall reply on the TCP port indicated in the M-SEARCH request, however it does not have to spread and repeat the required responses.

since the transport over TCP is reliable, hence ignoring the MX value. The reply to the control point can be formatted as 1 message replying to all applicable USN as required by the M-SEARCH syntax. The list of USNs can be conveyed by a comma separated list, see RFC 2616. Hence using this option will reduce the number of messages sent as responses to the M-SEARCH and will speed up the detection of devices in the network.

Any device responding to a unicast M-SEARCH should respond within 1 second.

The URL specified in the LOCATION header field of the [M-SEARCH](#) response shall be reachable by the control point to which the response is directed.

Responses to [M-SEARCH](#) requests are intentionally parallel to advertisements, and as such, follow the same pattern as listed for [NOTIFY](#) with [ssdp:alive](#) (above) except that instead of the [NT](#) header field there is an [ST](#) header field here. The response shall be sent in the following format. Values in *italics* are placeholders for actual values.

```
HTTP/1.1 200 OK
CACHE-CONTROL: max-age = seconds until advertisement expires
DATE: when response was generated
EXT:
LOCATION: URL for UPnP description for root device
SERVER: OS/version UPnP/2.0 product/version
ST: search target
USN: composite identifier for the advertisement
BOOTID.UPNP.ORG: number increased each time device sends an initial announce or an update message
CONFIGID.UPNP.ORG: number used for caching description information
SEARCHPORT.UPNP.ORG: number identifies port on which device responds to unicast M-SEARCH
```

Note: No body is present in a response to a request with method [M-SEARCH](#), but note that the message shall have a blank line following the last header field.

(Note: No need to limit TTL for the IP packet in response to a search request.)

Listed below are details for the header fields appearing in the listing above. Field names are not case sensitive. All field values are case sensitive except where noted.

Response line

Shall be "HTTP/1.1 200 OK"

Header fields

CACHE-CONTROL

Required. Field value shall have the max-age directive ("max-age=") followed by an integer that specifies the number of seconds the advertisement is valid. After this duration, control points should assume the device (or service) is no longer available; as long as a control point has received at least one advertisement that is still valid from a root device, any of its embedded devices or any of its services, then the control point can assume that all are available. The number of seconds should be greater than or equal to 1800 seconds (30 minutes), although exceptions are defined in the text above. Specified by UPnP vendor. Other directives shall not be sent and shall be ignored when received.

DATE

Recommended. Field value contains date when response was generated. "rfc1123-date" as defined in RFC 2616.

EXT

Required for backwards compatibility with UPnP 1.0. (Header field name only; no field value.)

LOCATION

Required. Field value contains a URL to the UPnP description of the root device. Normally the host portion contains a literal IP address rather than a domain name in unmanaged networks. Specified by UPnP vendor. Single absolute URL (see RFC 3986).

SERVER

Required. Specified by UPnP vendor. String. Field value shall begin with the following “product tokens” (defined by HTTP/1.1). The first product token identifies the operating system in the form *OS name/OS version*, the second token represents the UPnP version and shall be *UPnP/2.0*, and the third token identifies the product using the form *product name/product version*. For example, “SERVER: *unix/5.1 UPnP/2.0 MyProduct/1.0*”.

ST

Required. Field value contains Search Target. Single URI. The response sent by the device depends on the field value of the ST header field that was sent in the request. In some cases, the device shall send multiple response messages as follows. If the received ST field value was:

ssdp:all

Respond $3+2d+k$ times for a root device with d embedded devices and s embedded services but only k distinct service types (see clause 1.1.2, “SSDP message header fields” for a definition of each message to be sent). Field value for ST header field shall be the same as for the NT header field in NOTIFY messages with ssdp:alive. (See above.)

upnp:rootdevice

Respond once for root device. Shall be upnp:rootdevice.

uuid:*device-UUID*

Respond once for each matching device, root or embedded. Shall be uuid:*device-UUID* where *device-UUID* is specified by the UPnP vendor. See clause 1.1.4, “UUID format and recommended generation algorithms” for the MANDATORY UUID format.

urn:schemas-upnp-org:device:deviceType:ver

Respond once for each matching device, root or embedded. Shall be urn:schemas-upnp-org:device:deviceType:ver where *deviceType* and *ver* are defined by UPnP Forum working committee and *ver* shall contain the version of the device type contained in the M-SEARCH request.

urn:schemas-upnp-org:service:serviceType:ver

Respond once for each matching service type. shall be urn:schemas-upnp-org:service:serviceType:ver where *serviceType* and *ver* are defined by the UPnP Forum working committee and *ver* shall contain the version of the service type contained in the M-SEARCH request.

urn:*domain-name*:device:deviceType:ver

Respond once for each matching device, root or embedded. shall be urn:*domain-name*:device:deviceType:ver where *domain-name* (a Vendor Domain Name), *deviceType* and *ver* are defined by the UPnP vendor and *ver* shall contain the version of the device type from the M-SEARCH request. Period characters in the Vendor Domain Name shall be replaced with hyphens in accordance with RFC 2141.

urn:*domain-name*:service:serviceType:ver

Respond once for each matching service type. shall be urn: *domain-name*:service:serviceType:ver where *domain-name* (a Vendor Domain Name), *serviceType* and *ver* are defined by the UPnP vendor and *ver* shall contain the version of the service type from the M-SEARCH request. Period characters in the Vendor Domain Name shall be replaced with hyphens in accordance with RFC 2141.

USN

Required. Field value contains Unique Service Name. Identifies a unique instance of a device or service. shall be one of the following. (See Table 1 1, “Root device discovery messages”, Table 1 2, “Embedded device discovery messages”, and Table 1 3, “Service discovery messages” above.) The prefix (before the double colon) shall match the value of the UDN element in the device description OR the lower ST version used in the M-Search request. (Section 2, “Description” explains the UDN element.) Single URI.

uuid:*device-UUID*:upnp:rootdevice

Sent once for root device where *device-UUID* is specified by UPnP vendor. See section 1.1.4, “UUID format and Recommended generation algorithms” for the MANDATORY UUID format.

uuid:*device-UUID*

Sent once for every device, root or embedded, where device-UUID is specified by the UPnP vendor. See section 1.1.4, “UUID format and RECOMMENDED generation algorithms” for the MANDATORY UUID format.

uuid:device-UUID::urn:schemas-upnp-org:device:deviceType:ver

Sent once for every device, root or embedded, where device-UUID is specified by the UPnP vendor, deviceType and ver are defined by UPnP Forum working committee and ver specifies version of the device type. See section 1.1.4, "UUID format and RECOMMENDED generation algorithms" for the MANDATORY UUID format.

uuid:device-UUID::urn:schemas-upnp-org:service:serviceType:ver

Sent once for every service where device-UUID is specified by the UPnP vendor, serviceType and ver are defined by UPnP Forum working committee and ver specifies version of the device type. See section 1.1.4, "UUID format and RECOMMENDED generation algorithms" for the MANDATORY UUID format.

uuid:device-UUID::urn:domain-name:device:deviceType:ver

Sent once for every device, root or embedded, where device-UUID, domain-name (a Vendor Domain Name), deviceType and ver are defined by the UPnP vendor and ver specifies the version of the device type. See section 1.1.4, "UUID format and RECOMMENDED generation algorithms" for the MANDATORY UUID format. Period characters in the Vendor Domain Name shall be replaced by hyphens in accordance with RFC 2141.

uuid:device-UUID::urn:domain-name:service:serviceType:ver

Sent once for every service where device-UUID, domain-name (a Vendor Domain Name), serviceType and ver are defined by the UPnP vendor and ver specifies the version of the service type. See section 1.1.4, "UUID format and recommended generation algorithms" for the MANDATORY UUID format. Period characters in the Vendor Domain Name shall be replaced by hyphens in accordance with RFC 2141.

BOOTID.UPNP.ORG

Required. As defined in clause 1.2, and 1.2.2.

CONFIGID.UPNP.ORG

Allowed. As defined in clause 1.2, and 1.2.2.

SEARCHPORT.UPNP.ORG

Allowed. As defined in clause 1.2, and 1.2.2.

SECURELOCATION.UPNP.ORG

Allowed. As defined in clause 1.2.2.

If there is an error with the search request (such as an invalid field value in the MAN header field, a missing MX header field, or other malformed content), the device shall silently discard and ignore the search request; sending of error responses is PROHIBITED due to the possibility of packet storms if many devices send an error response to the same request.

1.4 References

RFC 2141, URN Syntax. Available at: <http://www.ietf.org/rfc/rfc2141.txt>.

RFC 2616, HTTP: Hypertext Transfer Protocol 1.1. Available at: <http://www.ietf.org/rfc/rfc2616.txt>.

RFC 2774, HTTP Extension Framework. Available at: <http://www.ietf.org/rfc/rfc2774.txt>.

RFC 3986, Uniform Resource Identifiers (URI): Generic Syntax. Available at: <http://www.ietf.org/rfc/rfc3986.txt>.

RFC 4340, Datagram Congestion Control Protocol (DCCP). Available at: <http://www.ietf.org/rfc/rfc4340.txt>.

[1] DCE variant of Universal Unique Identifiers (UUIDs), The Open group, 1997, Available at: <http://www.opengroup.org/onlinepubs/9629399/apdxa.htm>.

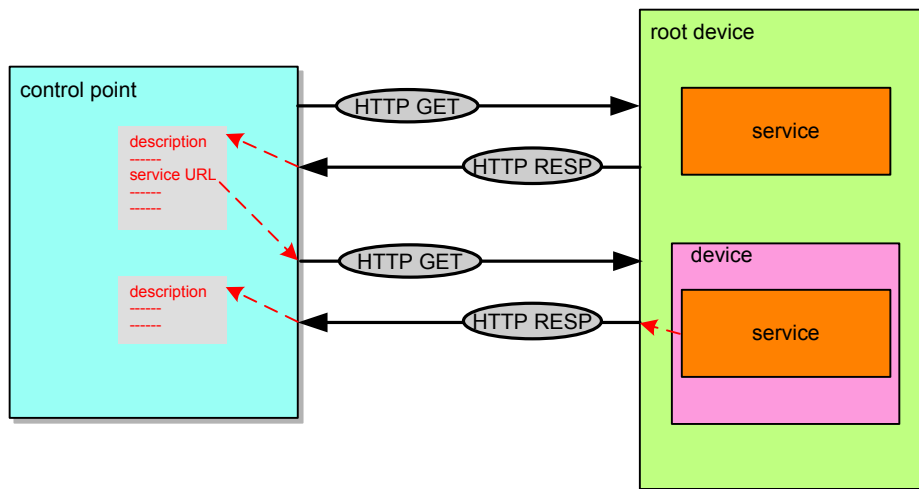
2 Description

Description is Step 2 in UPnP networking. Description comes after addressing (Step 0) where devices get a network address, and after discovery (Step 1) where control points find interesting device(s). Description enables control (Step 3) where control points send commands to device(s), eventing (Step 4) where control points listen to state changes.

device(s), and presentation (Step 5) where control points may display an html user interface for device(s).

After a control point has discovered a device, the control point still knows very little about the device -- only the information that was in the discovery message, i.e., the device's (or service's) UPnP type, the device's universally-unique identifier, and a URL to the device's UPnP description. For the control point to learn more about the device and its capabilities, or to interact with the device, the control point shall retrieve a description of the device and its capabilities from the URL provided by the device in the discovery message.

Figure 2-1: — Description architecture



The UPnP description for a device is partitioned into two logical parts: a *device description* describing the physical and logical containers, and *service descriptions* describing the capabilities exposed by the device. A UPnP device description includes vendor-specific manufacturer information like the model name and number, serial number, manufacturer name, URLs to vendor-specific Web sites, etc. (details below). For each service included in the device, the device description lists the service type, service name, a URL for a service description, a URL for control, and a URL for eventing. A device description also includes a description of all embedded devices and a URL for presentation of the aggregate. This clause explains UPnP device descriptions, and the clauses on Control, Eventing, and Presentation explain how URLs for control, eventing, and presentation are used respectively.

Note that a single physical device is allowed to include multiple logical devices. Multiple logical devices can be modeled as a single root device with embedded devices (and services) or as multiple root devices (perhaps with no embedded devices). In the former case, there is one UPnP device description for the root device, and that device description contains a description for all embedded devices. In the latter case, there are multiple UPnP device descriptions, one for each root device.

A UPnP device description is written by a UPnP vendor. The description is in XML syntax and is usually based on a standard UPnP Device Template. A UPnP Device Template is produced by a UPnP Forum working committee; they derive the template from the UPnP Device Schema, which was derived from standard constructions in XML. This clause explains the format for a UPnP device description, UPnP Device Templates, and the part of the UPnP Device Schema that covers devices.

A UPnP service description includes a list of commands, or *actions*, to which the service responds, and parameters, or *arguments* for each action. A service description also includes a list of variables. These variables model the state of the service at run time, and are described in terms of their data type, range, and event characteristics. This clause explains the description of actions, arguments, state variables, and the properties of those variables. The clause on Eventing explains event characteristics.

Like a UPnP device description, a UPnP service description is written by a UPnP vendor. The description is in XML syntax and is usually based on a standard UPnP Service Template. A UPnP Service Template is produced by a UPnP Forum working committee; they derived the template from the UPnP Service Schema, augmenting it with human language where necessary. The UPnP Service Schema is derived using the conventions of XML Schema. This clause explains the format for a UPnP service description, UPnP Service Templates, typical augmentations in human language, and the part of the UPnP Service Schema that covers services.

UPnP vendors can differentiate their devices by extending services (see clause 2.7, “Non-standard vendor extensions and limitations”), including additional UPnP services, or embedding additional devices. When a control point retrieves a particular device's description, these added features are exposed to the control point for control and eventing. The device and service descriptions authoritatively document the implementation of the device.

Retrieving a UPnP device description is simple: the control point issues an HTTP GET request on the URL in the discovery message, and the device returns the device description. Retrieving a UPnP service description is a similar process that uses a URL within the device description. The protocol stack, method, header fields, and body for the response and request are explained in detail below. Description documents shall be sent using the same IP address on which the HTTP GET request was received.

As long as *at least one* of the discovery advertisements from a root device, any of its embedded devices or any of its services have not expired and *none* of the advertisements have been cancelled, a control point is allowed to assume that the root device and all its embedded devices and all its services are available. The device and service descriptions are allowed to be retrieved at any point since the device and service descriptions are static as long as the device and its services are available. If a device cancels *at least one* of its advertisements or if *all* the advertisements expire, a control point should assume the device and its services are no longer available. If a device needs to change one of these descriptions, it shall cancel its outstanding advertisements and re-advertise. Consequently, control points should not assume that device and service descriptions are unchanged if a device re-appears on the network, but they can detect whether descriptions changed if a changed CONFIGID.UPNP.ORG field value is present in the announcements.

Like discovery, description plays an important role in the interoperability of devices and control points using different versions of UPnP networking. As explained in clause 1, “Discovery”, the UPnP Device Architecture is versioned with both a major and a minor version. The major version and minor version are separate integer numbers; they are not to be interpreted or compared as though they were a single decimal number, even though they are allowed to appear as such in print. Advances in minor versions shall be a compatible superset of earlier minor versions of the same major version; therefore device vendors are free to implement standardized devices and services on versions of the architecture with a higher minor version number. Advances in major version are not required to be supersets of earlier versions and are not guaranteed to be backward compatible. However UDA version 2.0 is specified as a superset of UDA 1.1 and is thus backwards compatible with UDA 1.x versions. Therefore UDA 2.0 control points shall maintain interoperability with UDA 1.x devices. UDA 1.x control points can work with UDA 2.0 devices, but can't access the additional functionality specified in UDA 2.0. The architecture version of a root device, all its embedded devices and all its services shall be the same. Version information is communicated in description messages as a backup to the information communicated in discovery messages. This clause explains the format of version information in description messages.

Device and service types standardized by UPnP Forum working committees or created by vendors have an integer version. Every later version of a device or service shall be a fully backwardly compatible superset of the previous version, i.e., compared to earlier versions of the device, it shall include all mandatory embedded devices and services of the same or later version. The UPnP device or service type remains the same across all versions of a device whereas the device or service version shall be larger for later versions. Versions of device and service templates are allowed to have non-integer versions (such as “0.9”) during development in the working committee, but this shall become an integer upon standardization. Devices and services are allowed to have a version number greater than the major version.