INTERNATIONAL STANDARD

ISO/IEC 29341-27-1

First edition
2017-09

Information technology — UPnP Device Architecture —

Part 27-1: Friendly device control protocol — Friendly information update service

Technologies de l'information — Architecture de dispositif UPnP —

Partie 27-1: Protocole de contrôle de dispositif convivial — Service de présence — Service convivial de mise à jour de l'information

Reference number
ISO/IEC 29341-27-1:2017(E)

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

Contents iii
List of Tables iv
List of Figures iv
1 Scope v
2 Introduction 1
3 Normative References 1
4 Terms, definitions, symbols and abbreviated terms 2
4.1 Provisioning terms 2
4.1.1 conditionally allowed 2
4.1.2 conditionally required 2
4.1.3 not allowed 2
4.2 Terms 2
4.2.1 *Clean* 2
4.2.2 *Pending* 2
4.2.3 *Friendly* 2
4.2.4 *Non-Restrictable* 2
4.2.5 *Restrictable* 2
4.3 Abbreviated terms 3
4.3.1 3
4.3.2 3
5 Notations and Conventions 3
5.1.1 Data Types 3
5.2 Management of XML Namespaces in Standardized DCPs 3
5.3 Vendor-defined Extensions 3
6 Service Modeling Definitions (Normative) 4
6.1 Service Type 4
6.2 Security Feature 4
6.2.1 Restrictable and Non-Restrictable actions 4
6.3 *FriendlyInfoUpdate* Service Architecture 5
6.4 State Variables 5
6.4.1 State Variable Overview 5
6.4.2 *FriendlyNameStatus* 5
6.4.3 *FriendlyIconListStatus* 6
6.4.4 *A_ARG_TYPE_NewName* 9
6.4.5 *A_ARG_TYPE_IconURI* 9
6.4.6 *A_ARG_TYPE_UpdateType* 9
6.4.7 *A_ARG_TYPE_Token* 9
6.4.8 *A_ARG_TYPE_RestoreType* 9
6.5 Eventing and Moderation 10
6.6 Actions 10
6.6.1 *GetFriendlyName()* 11
6.6.2 *GetFriendlyIconList()* 11
6.6.3 *SetFriendlyName()* 12

6.6.4 *SetFriendlyIconList()* 13
6.6.5 *RestoreFriendlyInfo()* 17
7 Theory of Operation (Informative) 18
7.1 Changing the device <friendlyName> 18
7.2 Changing the device <iconList> 19
8 XML Service Description 25

List of Tables

Table 6-1: Assignment of Restrictable/Non-Restrictable Roles 4
Table 6-2: State Variables 5
Table 6-3: AllowedValueList for *A_ARG_TYPE_UpdateType* 9
Table 6-4: AllowedValueList for *A_ARG_TYPE_RestoreType* 10
Table 6-5: Eventing and Moderation 10
Table 6-6: Actions 10
Table 6-7: Arguments for *GetFriendlyName()* 11
Table 6-8: Error Codes for *GetFriendlyName()* 11
Table 6-9: Arguments for *GetFriendlyIconList()* 12
Table 6-10: Error Codes for *GetFriendlyIconList()* 12
Table 6-11: Arguments for *SetFriendlyName()* 12
Table 6-12: Error Codes for *SetFriendlyName()* 13
Table 6-13: Specific behavior for *SetFriendlyIconList()* upon valid input 14
Table 6-14: Arguments for *SetFriendlyIconList()* 15
Table 2-16: Error Codes for *SetFriendlyIconList()* 16
Table 6-16: Arguments for *RestoreFriendlyInfo()* 17
Table 6-17: Error Codes for *RestoreFriendlyInfo()* 18

List of Figures

Figure 1 - Allowed `icon@status` transitions 16

Foreword

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

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

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

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

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

ISO/IEC 29341-27-1 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 WANPPPConnection:1 Service	ISO/IEC 29341-8-20
UPnP WLANConfiguration:1 Service	ISO/IEC 29341-8-21
UPnP WANDevice:1 Device	ISO/IEC 29341-8-3
UPnP WANConnectionDevice:1 Device	ISO/IEC 29341-8-4
UPnP WLANAccessPointDevice:1 Device	ISO/IEC 29341-8-5
UPnP Printer:1 Device	ISO/IEC 29341-9-1
UPnP ExternalActivity:1 Service	ISO/IEC 29341-9-10
UPnP Feeder:1.0 Service	ISO/IEC 29341-9-11
UPnP PrintBasic:1 Service	ISO/IEC 29341-9-12
UPnP Scan:1 Service	ISO/IEC 29341-9-13
UPnP Scanner:1.0 Device	ISO/IEC 29341-9-2
UPnP QoS Architecture:1.0	ISO/IEC 29341-10-1
UPnP QosDevice:1 Service	ISO/IEC 29341-10-10
UPnP QosManager:1 Service	ISO/IEC 29341-10-11
UPnP QosPolicyHolder:1 Service	ISO/IEC 29341-10-12
UPnP QoS Architecture:2	ISO/IEC 29341-11-1
UPnP QosDevice:2 Service	ISO/IEC 29341-11-10
UPnP QosManager:2 Service	ISO/IEC 29341-11-11
UPnP QosPolicyHolder:2 Service	ISO/IEC 29341-11-12
UPnP QOS v2 Schema Files	ISO/IEC 29341-11-2
UPnP RemoteUIClientDevice:1 Device	ISO/IEC 29341-12-1
UPnP RemoteUIClient:1 Service	ISO/IEC 29341-12-10
UPnP RemoteUIServer:1 Service	ISO/IEC 29341-12-11
UPnP RemoteUIServerDevice:1 Device	ISO/IEC 29341-12-2
UPnP DeviceSecurity:1 Service	ISO/IEC 29341-13-10
UPnP SecurityConsole:1 Service	ISO/IEC 29341-13-11
UPnP ContentDirectory:3 Service	ISO/IEC 29341-14-12:2011
UPnP MediaServer:3 Device	ISO/IEC 29341-14-3:2011
UPnP ContentSync:1	ISO/IEC 29341-15-10:2011

UPnP Low Power Architecture:1	ISO/IEC 29341-16-1:2011
UPnP LowPowerProxy:1 Service	ISO/IEC 29341-16-10:2011
UPnP LowPowerDevice:1 Service	ISO/IEC 29341-16-11:2011
UPnP QoS Architecture:3	ISO/IEC 29341-17-1:2011
UPnP QosDevice:3 Service	ISO/IEC 29341-17-10:2011
UPnP QosManager:3 Service	ISO/IEC 29341-17-11:2011
UPnP QosPolicyHolder:3 Service	ISO/IEC 29341-17-12:2011
UPnP QosDevice:3 Addendum	ISO/IEC 29341-17-13:2011
UPnP RemoteAccessArchitecture:1	ISO/IEC 29341-18-1:2011
UPnP InboundConnectionConfig:1 Service	ISO/IEC 29341-18-10:2011
UPnP RADAConfig:1 Service	ISO/IEC 29341-18-11:2011
UPnP RADASync:1 Service	ISO/IEC 29341-18-12:2011
UPnP RATAConfig:1 Service	ISO/IEC 29341-18-13:2011
UPnP RAClient:1 Device	ISO/IEC 29341-18-2:2011
UPnP RAServer:1 Device	ISO/IEC 29341-18-3:2011
UPnP RADiscoveryAgent:1 Device	ISO/IEC 29341-18-4:2011
UPnP SolarProtectionBlind:1 Device	ISO/IEC 29341-19-1:2011
UPnP TwoWayMotionMotor:1 Service	ISO/IEC 29341-19-10:2011
UPnP AV Architecture:2	ISO/IEC 29341-20-1
UPnP AVTransport:3 Service	ISO/IEC 29341-20-10
UPnP ConnectionManager:3 Service	ISO/IEC 29341-20-11
UPnP ContentDirectory:4 Device	ISO/IEC 29341-20-12
UPnP RenderingControl:3 Service	ISO/IEC 29341-20-13
UPnP ScheduledRecording:2 Service	ISO/IEC 29341-20-14
UPnP MediaRenderer:3 Service	ISO/IEC 29341-20-2
UPnP MediaServer:4 Device	ISO/IEC 29341-20-3
UPnP AV Datastructure Template:1	ISO/IEC 29341-20-4
UPnP InternetGatewayDevice:2 Device	ISO/IEC 29341-24-1
UPnP WANIPConnection:2 Service	ISO/IEC 29341-24-10
UPnP WANIPv6FirewallControl:1 Service	ISO/IEC 29341-24-11
UPnP WANConnectionDevice:2 Service	ISO/IEC 29341-24-2
UPnP WANDevice:2 Device	ISO/IEC 29341-24-3
UPnP Telephony Architecture:2	ISO/IEC 29341-26-1
UPnP CallManagement:2 Service	ISO/IEC 29341-26-10
UPnP MediaManagement:2 Service	ISO/IEC 29341-26-11
UPnP Messaging:2 Service	ISO/IEC 29341-26-12
UPnP PhoneManagement:2 Service	ISO/IEC 29341-26-13
UPnP AddressBook:1 Service	ISO/IEC 29341-26-14
UPnP Calendar:1 Service	ISO/IEC 29341-26-15
UPnP Presense:1 Service	ISO/IEC 29341-26-16
UPnP TelephonyClient:2 Device	ISO/IEC 29341-26-2
UPnP TelephonyServer:2 Device	ISO/IEC 29341-26-3
UPnP Friendly Info Update:1 Service	ISO/IEC 29341-27-1
UPnP MultiScreen MultiScreen Architecture:1	ISO/IEC 29341-28-1
UPnP MultiScreen Application Management:1 Service	ISO/IEC 29341-28-10
UPnP MultiScreen Screen:1 Device	ISO/IEC 29341-28-2
UPnP MultiScreen Application Management:2 Service	ISO/IEC 29341-29-10

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

1 Scope

This service definition is compliant with the UPnP Device Architecture version 1.0. It defines a service type referred to herein as FriendlyInfoUpdate service . It is scoped to the Device Description Document (DDD) and is a service allowing control points to create orderly updates to the `<friendlyName>` and `<iconList>` elements. Once a change has taken place the status of the DDD might not reflect that of the advertised description because the device can not leave the network during ongoing activities. Therefore a state variable is provided to indicate if the DDD contains non-advertised (*pending*) values. If for any reason the device goes off line, power cycles, or reboots it will most likely advertise its new DDD. If DeviceProtection UPnP DP is implemented on the device then it will support restricting control point actions to control points with specific Roles.

2 Introduction

The FriendlyInfoUpdate service is a UPnP service that enables control points to update specific, *friendly*, elements of a UPnP device's DDD. The DDD elements updateable by the service are:

- `<friendlyName>`
- `<iconList>`

The service also describes how to reset the value to the original factory setting and in the case of **`<iconList>`**, a method for updating the actual device icon binaries using HTTP methods. It is assumed that the device, when supporting the updating of icon binaries, will have the means to manage image binaries, such as determining MIME image type, height, width and color depth, as well as, negating undesired behavior associated with binary transfers involving potential malware.

3 Normative References

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

UPnP CDS4, UPnP ContentDirectory Service version 4.0 UPnP Forum, Available at http://upnp.org/specs/av/UPnP-av-ContentDirectory-v4-Service.pdf .

ISO/IEC 29341-1, UPnP Device Architecture, version 1.0, UPnP Forum. Available at: http://upnp.org/specs/arch/UPnPDA10_20000613.pdf Latest version available at: http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture.pdf

UPnP DP, UPnP DeviceProtection Service, version 1, UPnP Forum. Available at http://upnp.org/specs/gw/UPnP-gw-DeviceProtection-v1-Service.pdf .

RFC 2616, IETF RFC 2616, *HyperText Transport Protocol – HTTP/1.1*, R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, June 1999.
Available at: http://www.ietf.org/rfc/rfc2616.txt.

XML-FIS-SCHEMA, XML Schema for *FriendlyIconListStatus* state variable document. Available at: http://upnp.org/schemas/fd/fis-events.xsd.

XML-FNS-SCHEMA, – XML Schema for *FriendlyNameStatus* state variable document. Available at: http://upnp.org/schemas/fd/fns-events.xsd.

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

4 Terms, definitions, symbols and abbreviated terms

For the purposes of this document, the terms and definitions given in ISO/IEC 29341-1 and the following apply.

4.1 Provisioning terms

4.1.1 conditionally allowed

CA

The definition or behavior depends on a condition. If the specified condition is met, then the definition or behavior is allowed, otherwise it is not allowed.

4.1.2 conditionally required

CR

The definition or behavior depends on a condition. If the specified condition is met, then the definition or behavior is required, otherwise it is not allowed.

4.1.3 not allowed

The definition or behavior is prohibited by this specification. Opposite of required.

4.2 Terms

4.2.1 *Clean*

A notional device state scoped to this specification in which all internal states related to the DDD are identical to its last advertised Device Description Document.

4.2.2 *Pending*

The *pending* state is a notional device state scoped to this specification in which one or more of the internal states related to the DDD are different from its last advertised Device Description Document.

4.2.3 *Friendly*

DDD element(s) which can be interpreted as human identifiable and not tied directly to the device manufacturer information; specifically the `<friendlyName>` and device icons, defined in the `<iconList>` element, are considered *friendly*.

4.2.4 *Non-Restrictable*

A category of action, that when the DeviceProtection UPnP DP service is implemented on the device, cannot be blocked according to the presence or absence of a specific *Role* attached to a *Control Point Identity* or *User Identity*. See UPnP CDS4 for further explanation of *Role*, *Control Point Identity* and *User Identity*.

4.2.5 *Restrictable*

A category of actions that, when the DeviceProtection UPnP DP service is implemented on the device, can be blocked according to the presence or absence of a specific *Role* attached to a *Control Point Identity* or *User Identity*.

4.3 Abbreviated terms

4.3.1

A
allowed

4.3.2

DDD
device description document

5 Notations and Conventions

- Strings that are to be taken literally are enclosed in “double quotes”.
- Words that are emphasized are printed in *italic*.
- Keywords that are defined by the UPnP Working Committee are printed using the *forum* character style.
- Keywords that are defined by the UPnP Device Architecture are printed using the **arch** character style.
- A double colon delimiter, “::”, signifies a hierarchical parent-child (parent::child) relationship between the two objects separated by the double colon. This delimiter is used in multiple contexts, for example: Service::Action(), Action()::Argument, parentProperty::childProperty.

5.1.1 Data Types

This specification uses data type definitions from two different sources. The UPnP Device Architecture defined data types are used to define state variable and action argument data types ISO/IEC 29341-1. The XML Schema namespace is used to define property data types XML SCHEMA-2.

For UPnP Device Architecture defined Boolean data types, it is strongly recommended to use the value “**0**” for false, and the value “**1**” for true. The values “**true**”, “**yes**”, “**false**”, or “**no**” also allowed to be used but are not recommended. The values “**yes**” and “**no**” are deprecated and not allowed to be sent out by devices but shall be accepted on input.

For XML Schema defined Boolean data types, it is strongly recommended to use the value “*0*” for false, and the value “*1*” for true. The values “*true*”, “*yes*”, “*false*”, or “*no*” may also be used but are not recommended. The values “*yes*” and “*no*” are deprecated and not allowed to be sent out by devices but shall be accepted on input.

5.2 Management of XML Namespaces in Standardized DCPs

This specification shall follow the conventions of the equivalent section of UPnP CDS4 where applicable.

5.3 Vendor-defined Extensions

Whenever vendors create additional vendor-defined state variables, actions or properties, their assigned names and XML representation shall follow the naming conventions and XML rules as specified in ISO/IEC 29341-1, Section 2.5, “Description: Non-standard vendor extensions”.

6 Service Modeling Definitions (Normative)

6.1 Service Type

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

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

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

6.2 Security Feature

In the specification, if support for the *Security Feature* is referenced, this indicates that the device implements the DeviceProtection Service UPnP DP.

6.2.1 Restrictable and Non-Restrictable actions

The FriendlyInfoUpdate service actions defined in this specification have the *Restrictable*, *Non-Restrictable* assignments as indicated in Table 6-1.

Table 6-1: Assignment of Restrictable/Non-Restrictable Roles

Action Name	*Restrictable/Non-Restrictable* to indicated *Role*[1]			
	Public	*Basic*	*Admin*	*Vendor Defined*
GetFriendlyName()	NO	NO	NO	NO
GetFriendlyIconList()	NO	NO	NO	NO
SetFriendlyName()	YES	YES	NO	YES
SetFriendlyIconList()	YES	YES	NO	YES
RestoreFriendlyInfo()	YES	YES	NO	YES

[1] A YES value in the table indicates that the action shall be *Restrictable* when the *Security Feature* is supported and a control point with only the *Role* indicated shall not have *Action level access*, (see UPnP CDS4) and shall receive an error code 606 (see UPnP Device Architecture ISO/IEC 29341-1) in response to the action invocation.

A NO value in the table indicates that the action shall be *Non-Restrictable*, meaning that even if the *Security Feature* is supported all control points shall have *Action level access* when control point invoking the action and shall not receive an error code 606 based on the *Security Feature*.

6.3 *FriendlyInfoUpdate* Service Architecture

This service is provided by UPnP devices to allow UPnP control points to get (read) and set (write) the value of the `<friendlyName>` and `<iconList>` elements of the UPnP Device Description Document (DDD). The `<friendlyName>` element is required and its value can be found in the UPnP Device Description Document made available in the description phase. The `<iconList>` element is only required if one or more `<icon>` elements are included in the DDD. In most cases, without this service, a UPnP control point is not able to change the device's `<friendlyName>` or `<iconList>`, however if there are other methods available, this service also defines how interaction with the FriendlyInfoUpdate service state variables is handled.

This service enables control points to get the most current <friendlyName> and <iconList> and set them to more informative and intuitive versions such as "John's living room TV" instead of the default "X company Media Renderer model 123".

The service enforces some basic timing restrictions to prevent race conditions when multiple updates are requested in quick succession or multiple control points are interacting with the service.

6.4 State Variables

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

6.4.1 State Variable Overview

Table 6-2: State Variables

Variable Name	R/A[1]	Data Type	Reference
FriendlyNameStatus	*R*	**string**	See Section 6.4.2
FriendlyIconListStatus	*CR*	**string**	See Section 6.4.3
A_ARG_TYPE_NewName	*R*	**string**	See Section 6.4.4
A_ARG_TYPE_IconURI	*CR*	**string**	See Section 6.4.5
A_ARG_TYPE_UpdateType	*CR*	**string**	See Section 6.4.6
A_ARG_TYPE_Token	*CR*	**string**	See Section 6.4.7
A_ARG_TYPE_RestoreType	*CR*	**string**	See Section 6.4.8

[1] *R* = Required, *A* = Allowed, *CR* = Conditionally Required, *CA* = Conditionally Allowed, *X* = Non-standard, add *-D* when deprecated (e.g., *R-D*, *A-D*).

6.4.2 *FriendlyNameStatus*

This required state variable indicates the status of the DDD `<friendlyName>` element value relative to changes incurred via the FriendlyInfoUpdate service or any other method. If the current `<friendlyName>` element value is different from the last `<friendlyName>` element value advertised in the DDD then the `<friendlyName> @status` attribute shall have a value of "*DDD*", otherwise it shall have a value of "*PENDING*". If the DDD `<friendlyName>` element value is changed by any other method prior to it being re-advertised in the DDD then the new value shall also be reflected in the *FriendlyNameStatus* state variable with the appropriate `@status` value.

The *FriendlyNameStatus* state variable is a string that contains a *FriendlyNameStatus XML Document*. The following example shows a generalized "template" for the *FriendlyNameStatus XML Document*. The example shows the elements that need to be filled out by individual implmentations in the *vendor* character style. The Schema associated with the *FriendlyNameStatus* state variable is can be found at XML-FNS-SCHEMA.

```
<?xml version="1.0" encoding="UTF-8"?>
<FriendlyNameStatus
   xmlns="urn:schemas-upnp-org:fd:fns-events"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:fd:fns-events
   http://www.upnp.org/schemas/fd/fns-events.xsd">
   <friendlyName status="DDD|PENDING">
      Describes the current or pending value for the DDD friendlyName
   </friendlyName>
</FriendlyNameStatus>
```

<xml>

Allowed. Case sensitive.

<FriendlyNameStatus>

Required. <XML>. Shall include a namespace declaration for the FriendlyInfoUpdate service Common Datastructures Schema ("urn:schemas-upnp-org:fd:fns-events"). This is the base wrapper element for the state variable. Shall include one of the following elements:

<friendlyName>

Required. xsd:string. Shall appear one time. The value of this element is the same as defined in the <friendlyName> element as defined in ISO/IEC 29341-1. The value of this element shall not be empty.

@status

Required. xsd:string. Shall be one of the following values: "*DDD*" or "*PENDING*". "*DDD*" indicates that the **<friendlyName>** is the same as advertised in the DDD. "*PENDING*" indicates that the **<friendlyName>** value has changed and is awaiting re-advertisement of the DDD.

6.4.3 *FriendlyIconListStatus*

This conditionally required state variable indicates the status of the DDD <iconList> element relative to changes incurred via the FriendlyInfoUpdate service or any other method. If the device is capable of advertising an <iconList> element in its DDD then it shall support this state variable, otherwise it is not allowed. It shall provide the complete internal <iconList> relative to the advertised DDD <iconList> and is allowed to expose additional resources for creating new icons via an <icon> element <postToken> or <getToken> element. If the DDD <iconList> element value is changed by any other method prior to being re-advertised in the DDD then the new value shall also be reflected in the *FriendlyIconListStatus* state variable. Note that the *FriendlyIconListStatus* state variable shall always have an <iconList> element present and should have at least one <icon> element with <mimetype> element value "image/png" since this is recommended in ISO/IEC 29341-1.

The *FriendlyIconListStatus* state variable is a string that contains a *FriendlyIconListStatus XML Document*. The following example shows a generalized "template" for the *FriendlyIconListStatus XML Document*. The example shows the elements that need to be filled out by individual implementations in the *vendor* character style. The Schema associated with the *FriendlyIconListStatus* state variable is can be found at XML-FIS-SCHEMA.

```
<?xml version="1.0" encoding="UTF-8"?>
<FriendlyIconListStatus
   xmlns="urn:schemas-upnp-org:fd:fis-events"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:fd:fis-events
   http://www.upnp.org/schemas/fd/fis-events.xsd">
   <iconList>
      !-- Describes current, complete internal device <iconList>.
```



```
        Includes current DDD icons, pending updates and deletions,
       and resources to create new icons -->
       <icon status="DDD|DELETED|OPEN|PENDING"
         maxbyte="maximum size of icon in bytes"
         maxheigth="maximum heigth of icon in pixels"
         maxwidth="maximum width of icon in pixels"
         maxdepth="maximum color depth of icon in bits">
          <url>URL to existing, deleted, or pending icon binary</url>
          <mimetype>image/format</mimetype>
          <height>vertical pixels</height>
          <width>horizontal pixels</width>
          <depth>color depth</depth>
          <getToken>
             a device unique tag used to trigger a device HTTP-GET
             of a supplied icon binary resource, only used for replacing
             or creating an icon binary
          </getToken>
          <postToken postUri="valid url for posting a new icon binary">
             a device unique tag used by a control point to authorize
             an HTTP-POST (upload) of an icon binary, only used
             for replacing or creating an icon binary
          </postToken>
       </icon>
    </iconList>
</FriendlyIconListStatus>
```

<xml>

Allowed. Case sensitive.

<FriendlyIconListStatus>

Required. <XML>. Shall include a namespace declaration for the FriendlyInfoUpdate service Common Datastructures Schema ("urn:schemas-upnp-org:fd:fis-events"). This is the base wrapper element for the state variable. Shall include one of the following elements:

<iconList>

Required. <XML>. Shall appear one time. The contents of this element shall contain one or more of the following elements:

<icon>

Required. <XML>. Shall appear one or more times. Describes an existing icon (either *clean* or *pending*) and its associated characteristics, a deleted (*pending*) icon, or a URI resource where an icon can be created.

@status

Required. xsd:string. Indicates the status of the icon. Shall be one of the following values: "*DDD*", "*DELETED*", "*PENDING*," "*OPEN*".

"*DDD*" indicates that the **<icon>** is part of the advertised DDD and the icon binaries are the same.
"*DELETED*" indicates that the **<icon>** is currently part of the advertised DDD but the <icon> and its binary are pending deletion.
"*OPEN*" indicates that the **<icon>** is a place holder for the addition of a new <icon> with a new icon binary.
"*PENDING*" indicates that the **<icon>** has been created from a previously OPEN **<icon>** slot, its binary is available and it is pending advertisement in the DDD when it is updated.

@maxBytes

Conditionally allowed. xsd:unsignedint. Shall appear zero or one time. Indicates the maximum size (in bytes) for a new icon binary. The value should be on a per icon basis especially if the **@maxheight**, **@maxwidth** or **@maxdepth** attributes are also present. If, on the otherhand, it is a device level limitation (and not an individual icon limitation) then it should be uniformly updated across all icons with an **@status** value of "*OPEN*" when a binary is uploaded. Shall only be present if the parent <icon> has **@status** attribute value of "*OPEN*", otherwise it is not allowed.

@maxHeight

Conditionally allowed. xsd:unsignedint. Shall appear zero or one time. Indicates the maximum vertical size (in pixels) for a new icon binary. Shall only be present if the parent <icon> has **@status** attribute value of "*OPEN*", otherwise it is not allowed.

@maxWidth
Conditionally allowed. xsd:unsignedint. Shall appear zero or one time. Indicates the maximum horizontal size (in pixels) for a new icon binary. Shall only be present if the parent <icon> has **@status** attribute value of "*OPEN*", otherwise it is not allowed.

@maxDepth
Conditionally allowed. xsd:unsignedint. Shall appear zero or one time. Indicates the maximum color depth (in bits) for a new icon binary. Shall only be present if the parent <icon> has **@status** attribute value of "*OPEN*", otherwise it is not allowed.

<url>
Conditionally allowed. xsd:anyURI. Shall appear zero or one time. Indicates a URI where an icon binary of the MIME image type indicated in the sibling element **<mimetype>** exists. Shall be present if the icon is part of the current DDD (**@status** of "*DDD*"), has been created and is waiting advertisement in the DDD (**@status** of "*PENDING*"), or is present in the current DDD and pending deletion (**@status** of "*DELETED*"). Shall not be present if the icon is awaiting creation (**@status** of "*OPEN*").

<postToken>
Conditionally allowed. xsd:string. Shall appear zero or one time. Provides a device unique token which can be passed to the device by the control point to authorize an HTTP-POST to the device to the URI provided by the **postUri@iconToken**. The posted binary shall be of the MIME image type indicated in the sibling element **<mimetype>**. Shall only be present if the device supports the HTTP-POST operation and the **@status** attribute has a value of "*OPEN*".

@postUri
Required. xsd:anyURI. Indicates a resource that can be used to support an HTTP-POST RFC 2616 for a new ("*OPEN*") icon binary. Note that full and relative URIs are allowed.

<getToken>
Conditionally Required. xsd:string. Shall appear one time. Provides a device unique token which can be passed to the device by the control point to trigger an HTTP-GET by the device for the parent **<icon>**. The acquired icon binary shall be of the MIME image type indicated in the sibling element **<mimetype>**. Shall be present if the parent <icon> has **@status** attribute value of "*OPEN*", otherwise it is not allowed.

<mimetype>
Required. xsd:string. Shall appear one time. Indicates the image MIME image type of the icon binary that shall be associated with the parent **<icon>** element. If the **<icon>** element has an **@status** value of "*OPEN*" then the value indicates the MIME image type of the image binary that can be uploaded, otherwise the value indicates the MIME image type of an existing icon binary (in the *clean* or *pending* state). If multiple MIME image type binaries are supported by the device then the device shall use a different **<icon>** with the appropriate **<mimetype>** element for each MIME image type supported.

<width>
Conditionally required. xsd:int. Shall appear zero or one time. Indicates the horizontal dimension (in pixels) of the icon binary associated with the parent **<icon>** element. Shall be present if the icon **@status** attribute is "*DDD*" or "*PENDING*". Shall not be present if the icon **@status** attribute is "*DELETED*" or "*OPEN*".

<height>
Conditionally required. xsd:int. Shall appear zero or one time. Indicates the vertical dimension (in pixels) of the icon binary associated with the

parent **<icon>** element. Shall be present if the icon **@status** attribute is "*DDD*" or "*PENDING*". Shall not be present if the icon **@status** attribute is "*DELETED*" or "*OPEN*".

<depth>
Conditionally required. xsd:int. Shall appear zero or one time. Indicates the color bits in the icon binary associated with the parent **<icon>** element. Shall be present if the icon **@status** attribute is "*DDD*" or "*PENDING*". Shall not be present if the icon **@status** attribute is "*DELETED*" or "*OPEN*".

Note that the availability of the <postToken> is dependent on the device supporting the HTTP-POST method. The device shall support the <postToken> resource uniformly on all creatable icons in the *FriendlyIconListStatus* state variable with the @status attribute of "*OPEN*", that is, it shall have a <postToken> on all "*OPEN*" icons, not a mix. Note that this is addressed for <getToken> by its conditionally required status.

6.4.4 *A_ARG_TYPE_NewName*

This required state variable provides type information for the *NewName* argument in the *SetFriendlyName()* action. The data type is **string**.

6.4.5 *A_ARG_TYPE_IconURI*

This conditionally required state variable shall be present if the *SetFriendlyIconList()* action is supported otherwise it is not allowed. It provides type information for the *TargetIconURI* argument in the *SetFriendlyIconList()* action. The data type is **string**.

6.4.6 *A_ARG_TYPE_UpdateType*

This conditionally required state variable shall be present if the *SetFriendlyIconList()* action is supported otherwise it is not allowed. It provides type information for the *Method* argument in the *SetFriendlyIconList()* action. The data type is **string**.

Table 6-3: AllowedValueList for *A_ARG_TYPE_UpdateType*

Value	R/A	Description
CREATE	*R*	Indicates that the control point is requesting that a new icon (with a new icon binary) be created in an icon slot with @status value of "*OPEN*".
DELETE	*R*	Indicates that the control point is requesting that an existing icon binary with @status value of "*DDD*" or "*PENDING*" be deleted.

6.4.7 *A_ARG_TYPE_Token*

This conditionally required state variable shall be present if the *SetFriendlyIconList()* action is supported otherwise it is not allowed. It provides type information for the *TargetIconToken* input argument in the *SetFriendlyIconList()* action. The data type is **string**.

6.4.8 *A_ARG_TYPE_RestoreType*

This conditionally required state variable shall be present if the *RestoreFriendlyInfo()* action is supported, otherwise it is not allowed. It provides type information for the *RestoreType* argument in the *RestoreFriendlyInfo()* action. The data type is **string**.

Table 6-4: AllowedValueList for *A_ARG_TYPE_RestoreType*

Value	R/A	Description
ALL	*CR*	Indicates that the control point is requesting that all of the device *friendly* DDD information be restored to its factory settings. Shall be supported if the *SetFriendlyIconList* action is supported, otherwise it is not allowed.
FRIENDLYNAME	*R*	Indicates that the control point is requesting that a device `<friendlyName>` be restored to its factory setting.
ICONLIST	*CR*	Indicates that the control point is requesting that a device `<iconList>` be restored to its factory setting. Shall be supported if the *SetFriendlyIconList* action is supported, otherwise it is not allowed.

6.5 Eventing and Moderation

Table 6-5: Eventing and Moderation

		Moderation	
Variable Name	Evented	Moderated[1]	Criteria
FriendlyNameStatus	*Yes*	*No*	
FriendlyIconListStatus	*Yes*	*No*	
A_ARG_TYPE_NewName	*No*	*No*	
A_ARG_TYPE_IconURI	*No*	*No*	
A_ARG_TYPE_UpdateType	*No*	*No*	
A_ARG_TYPE_Token	*No*	*No*	
A_ARG_TYPE_RestoreType	*No*	*No*	

[1] *YES* = The state variable shall be moderated with the criteria.

6.6 Actions

Table 6-6: Actions

Name	Device R/A[1]	Control Point R/A[2]
GetFriendlyName()	*R*	*R*
GetFriendlyIconList()	*CR*	*A*
SetFriendlyName()	*R*	*R*
SetFriendlyIconList()	*CR*	*A*
RestoreFriendlyInfo()	*A*	*A*

[1] For a device this column indicates whether the action shall be implemented or not, where *R* = Required, *A* = Allowed, *CR* = Conditionally Required, *CA* = Conditionally Allowed, *X* = Non-standard, add *-D* when deprecated (e.g., *R-D*, *O-D*).

2 For a control point this column indicates whether a control point shall be capable of invoking this action, where *R* = Required, *A* = Allowed, *CR* = Conditionally Required, *CA* = Conditionally Allowed, *X* = Non-standard, add *-D* when deprecated (e.g., *R-D*, *O-D*).

6.6.1 *GetFriendlyName()*

This required action is used to get the internal value of the `<friendlyName>` element (*pending* or *clean*) of the *FriendlyNameStatus* state variable. This status is returned in the *NameStatus* output argument which contains a properly escaped *FriendlyNameStatus XML Document*.

6.6.1.1 Arguments

Table 6-7: Arguments for *GetFriendlyName()*

Argument	Direction	relatedStateVariable
NameStatus	*OUT*	*FriendlyNameStatus*

6.6.1.2 Device Requirements

None.

6.6.1.3 Dependency on State (if any)

Return value is dependent on value of the *FriendlyNameStatus* state variable. If the *Security Feature* is supported the action is defined as a *Non-Restrictable* action and the targeted device is not allowed to restrict its invocation by any control point.

6.6.1.4 Effect on State (if any)

None.

6.6.1.5 Errors

Table 6-8: Error Codes for *GetFriendlyName()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
700		Reserved for future extensions.

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

6.6.2 *GetFriendlyIconList()*

This conditionally required action is used to get any current *clean* or *pending* values of the `<iconList>` element in the DDD via the *FriendlyIconListStatus* state variable. This status is returned in the *IconListStatus* output argument which contains the a properly escaped

FriendlyIconListStatus XML Document. If the device is capable of advertising an `<iconList>` element in its DDD then it shall support this action otherwise it is not allowed.

6.6.2.1 Arguments

Table 6-9: Arguments for *GetFriendlyIconList()*

Argument	Direction	relatedStateVariable
IconListStatus	*OUT*	*FriendlyIconListStatus*

6.6.2.2 Device Requirements

None.

6.6.2.3 Dependency on State (if any)

Return value is dependent on value of the *FriendlyIconListStatus* state variable. If the *Security Feature* is supported the action is defined as a *Non-Restrictable* action and the targeted device is not allowed to restrict its invocation by any control point.

6.6.2.4 Effect on State (if any)

None.

6.6.2.5 Errors

Table 6-10: Error Codes for *GetFriendlyIconList()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
700		Reserved for future extensions.

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

6.6.3 *SetFriendlyName()*

This required action is used to set a new value to the `<friendlyName>` element in the DDD. The *NewName* input argument identifies the requested new `<friendlyName>` value. If the *NewName* input argument value is empty (blank) the the action shall return errorCode 702.

6.6.3.1 Arguments

Table 6-11: Arguments for *SetFriendlyName()*

Argument	Direction	relatedStateVariable
NewName	*IN*	*A_ARG_TYPE_NewName*

6.6.3.2 Device Requirements

If the device supports the *Security Feature* then this action is defined as a *Restrictable* action and the targeted device is allowed to restrict its invocation to specific control points which are identifiable or have specific *Roles*.

6.6.3.3 Dependency on State (if any)

None.

6.6.3.4 Effect on State (if any)

Successful invocation of this action will change the value of the *FriendlyNameStatus* state variable and cause it to be evented. Unsuccessful invocation of this action, that is the return of any errorCode except 709 shall also cause the *FriendlyNameStatus* state variable to be evented, however it will be unchanged. This is done to indicate to other control points that the 30 second (or less) lock-out window has expired. If the device is in a state that allows it to leave the network and re-advertise its new friendly values then it should do so, however, it shall wait at least 30 seconds after any invocation of the *GetFriendlyName()*, *GetFriendlyIconList()*, *SetFriendlyName()*, *SetFriendlyIconList()*, or *RestoreFriendlyInfo()* actions before initiating the re-advertisement behavior; this is done to provide opportunities for control points to make more than one change to the the DDD without a re-advertisement.. The timing for leaving and re-advertising is implementation dependent but shall be confomant to ISO/IEC 29341-1. Once the device re-advertises the value of its DDD `<friendlyName>` element it should be the same as the value of the *FriendlyNameStatus* state variable `<friendlyName>` element immediately prior to the device leaving the network and the *FriendlyNameStatus* state variable `<friendlyName>` element should be in the *clean* state, that is, its `@status` value should be "*DDD*".

6.6.3.5 Errors

Table 6-12: Error Codes for *SetFriendlyName()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
700		Reserved for future extensions.
701	Name too long	The provided friendlyName value is too long.
702	Empty name not allowed	The friendlyName value shall not have a non-empty (blank) name.

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

6.6.4 *SetFriendlyIconList()*

This conditionally required action is used to create a new icon or delete an existing *clean* or *pending* icon for use in a *friendly* element `<iconList>` of the DDD. If the device supports the *FriendlyIconListStatus* state variable then this action is required otherwise it is not allowed. It acts on information provided in the `<iconList>` element of *FriendlyIconListStatus* state variable, which reflects clean, pending, and potentially new icon information.

The *UpdateType* input argument indicates the direction of the action, that is, whether the actions outcome is to create a new icon or delete an existing icon.The *TargetIconToken* input argument ties together a specific `<icon>` element and associated `<url>` element with a specific HTTP (GET or POST) operation.

In the case of an "*CREATE*" action, the *TargetIconURI* input argument indicates where the primary binary resource of the HTTP-GET operation is located. It is allowed be internal or external to the device; for example the device is also a media server and the control point wants to use an exposed image item as an icon.

In the case of a "*DELETE*" action, the *TargetIconURI* input argument indicates the internal `<url>` of the icon binary resource to be deleted.

The behavior of the *SetFriendlyIconList()* action is further described in Table 6-13 for valid input combinations of the *UpdateType*, *TargetIconToken*, and *TargetIconURI* input arguments. These are the only valid combinations of input arguments that result in a successful icon update.

Table 6-13: Specific behavior for *SetFriendlyIconList()* upon valid input

UpdateType	*TargetIconToken*	*TargetIconURI*	Supported Operation[1]
"*CREATE*"	Matches an existing *FriendlyIconListStatus* state variable `<icon>` element `<postToken>` element with associated `@status` value of "*OPEN*"	Empty	The device shall immediately allow an HTTP-POST to the `postUri@postToken` associated with the *TargetIconURI* input for 30 seconds.
	Matches an existing *FriendlyIconListStatus* state variable `<icon>` element `<getToken>` element with associated `@status` value of "*OPEN*"	Shall be a valid URI	The device shall attempt and complete an HTTP-GET of *TargetIconURI* within 30 seconds.
"*DELETE*"	Empty	Matches an existing *FriendlyIconListStatus* state variable `<url>` element with associated `@status` value of "*PENDING*" or "*DDD*".	The device should mark for deletion the image binary identified by the *TargetIconURI* input argument within 30 seconds, unless the device implementation needs to persist the targeted icon.[2]

Upon invocation of the *SetFriendlyIconList()* action, the device shall determine if the input arguments are a valid combination or return an errorCode 703.

Upon invocation of the *SetFriendlyIconList()* action, this action shall complete within 30 seconds, including the requested HTTP-GET or HTTP-POST, or return an errorCode 704, unless one of the other errorCodes are returned prior to the 30 second window.

The device shall prohibit HTTP-POST to any `<postUri@postToken>` resource advertised in the *FriendlyIconListStatus* state variable unless a *SetFriendlyIconList()* action has been invoked on a specific `<postUri@postToken>`, then it shall be allowed on that resource for no more than 30 seconds.

The device shall not issue an HTTP-GET for any `<icon>` resources advertised in the *FriendlyIconListStatus* state variable unless a *SetFriendlyIconList()* action has been invoked on the specific `<icon>`, then it shall do so on that resource within 30 seconds.

If the action requests an HTTP-POST or HTTP-GET for any `<icon>` and the MIME image type of the image binary does not match the `<mimetype>` element then the action shall return errorCode 705.

[1] Upon successful completion, the associated *FriendlyIconListStatus* state variable `<icon>` element shall be updated according to the new icon binaries and the *FriendlyIconListStatus* state variable evented.

[2] In some cases the device implementation may want to retain certain icons, at least until a suitable replacement icon has been created, this behavior is allowed.

The device shall not expose values associated with the *FriendlyIconListStatus* state variable `<url>`, `<mimetype>`, `<height>`, `<width>`, or `<depth>` elements inconsistent with ISO/IEC 29341-1 (see section 6.6.4.4 for additional detail).

The device is allowed to fail the action and return errorCode 706 through709 if no other errorCodes are required as previously indicated.

The device is allowed to add or remove `<icon>` elements with the `@status` value of "*OPEN*" upon any eventing of the *FriendlyIconListStatus* state variable, that is, the device is allowed to add or remove resources dynamically for creating new icons dependent on its specific capabilities.

6.6.4.1 Arguments

Table 6-14: Arguments for *SetFriendlyIconList()*

Argument	Direction	relatedStateVariable
UpdateType	*IN*	*A_ARG_TYPE_UpdateType*
TargetIconToken	*IN*	*A_ARG_TYPE_Token*
TargetIconURI	*IN*	*A_ARG_TYPE_IconURI*

6.6.4.2 Device Requirements

If the device supports the *Security Feature* then this action is defined as a *Restrictable* action and the targeted device is allowed to restrict its invocation to specific control points which are identifiable or have specific *Roles*.

6.6.4.3 Dependency on State (if any)

If the targeted *friendly* element has been changed then the execution is dependent on the current *FriendlyIconListStatus* state variable as described above. Prior to re-advertisment of the DDD, transitions of `<icon>` `@status` from the "*DDD*","*DELETED*","*PENDING*", and "*OPEN*" states are restricted as follows:

- An `<icon>` with `@status` of "*DDD*" shall only transition to an `@status` value of "*DELETED*". Restoration of the original binary can be achieved in two ways: 1) reloading the original binary in a "*OPEN*" icon slot, 2) a successful invocation of the *RestoreFriendlyInfo()* action.
- An `<icon>` with `@status` of "*OPEN*" shall only transition to an `@status` value of "*PENDING*".
- An `<icon>` with `@status` of "*PENDING*" shall only transition to an `@status` value of "*OPEN*" if the icon binary is deleted.

6.6.4.4 Effect on State (if any)

If the invocation of the *SetFriendlyIconList()* action creates a binary for the icon associated with the *TargetIconToken* input argument having an `@status` value of "*OPEN*" then the device shall:

1) Update the corresponding `<url>`, `<width>`, `<height>`, `<depth>`, and `<mimetype>` values of the *FriendlyIconListStatus* state variable to correctly indicate the new image characteristics.
2) Change the `<icon>` `@status` value to "*PENDING*".

If the invocation of the *SetFriendlyIconList()* action deletes the binary of the icon indicated by the *TargetIconURI* input argument having an `@status` value of "*DDD*" or "*PENDING*" then the device shall update the *FriendlyIconListStatus* state variable as follows:

1) Remove the associated `<width>`, `<height>`, and `<depth>` elements to correctly indicate the impending lack of an icon binary.

2) Change the associated `@status` attribute value to "*DELETED*" if the icon is currently part of the advertised DDD or "*OPEN*" if it had a previous `@status` value of "*PENDING*".

3) Remove the `<url>` element, if the icon transitions to an `@status` value of "*OPEN*", and add the `<getToken>`, `<postToken>` or both. The `<postToken>` element shall be added in a uniform manner as previously described.

4) Add the <maxBytes>, <maxHeight>, <maxWidth>, and <maxDepth> elements if supported.

The device shall event the *FriendlyIconListStatus* state variable upon any change. Unsuccessful invocation of this action, that is the return of any errorCode except 709, shall also cause the *FriendlyIconListStatus* state variable to be evented, however it will be unchanged. This is done to indicate to other control points that the 30 second (or less) lock-out window has expired.

The allowed state transitions are illustrated in Figure 1 along with the FriendlyInfoUpdate service actions (*SetFriendlyName()* and *RestoreFriendlyInfo()*) or a DDD re-advertisement that can trigger the state change.

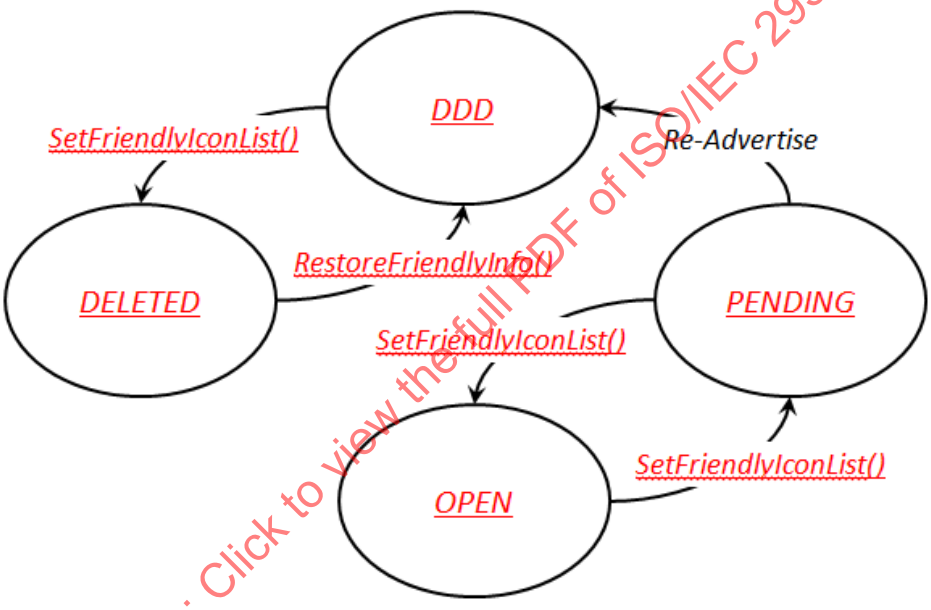

Figure 1 - Allowed `icon@status` transitions

If the device is in a state that allows it to leave the network and re-advertise its new *friendly* values then it should do so however it shall wait at least 30 seconds after any invocation of any *GetFriendlyName()*, *GetFriendlyIconList()*, *SetFriendlyName()*, *SetFriendlyIconList()* or *RestoreFriendlyInfo()* actions before initiating the re-advertisement behavior; this is done to provide opportunities for control points to make more than one change to the the DDD without a re-advertisement. The timing for leaving and re-advertising is implementation dependent but shall be confomant to ISO/IEC 29341-1.

6.6.4.5 Errors

Table 6-15: Error Codes for *SetFriendlyIconList()*

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

ErrorCode	errorDescription	Description
600-699	TBD	See UPnP Device Architecture section on Control.
700		Reserved for future extensions.
703	Invalid Input	The combination of input parameters is not allowed.
704	HTTP timeout	The HTTP-POST or HTTP-GET for the icon resource did not complete in the allowed time.
705	MIME image type mismatch	The icon binary MIME image type is not supported by this device.
706	Delete not allowed.	A delete was attempted on an icon with an `@status` of "*OPEN*" or "*DELETED*" or an icon with an `@status` of "*DDD*" that the device is retaining.
707	Unknown binary	The HTTP-GET binary resource cannot be found.
708	Binary Rejected	The device cannot support an icon binary of this size. This error code can be returned for image binaries that are not constrained to any exposed limitation via the `@maxBytes`, `@maxHeight`, `@maxWidth` or `@maxDepth` attributes.
709	Icon Update in Progress	The device is currently executing an Icon update and is in the 30 second lock-out period.

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

6.6.5 *RestoreFriendlyInfo()*

This allowed action is used to restore the `<friendlyName>` and, if supported, the `<iconList>` element in the DDD to its factory default settings. The *RestoreType* input argument identifies if the request is to restore the `<friendlyName>` value or the `<iconList>` values or both.

If the *RestoreType* input parameter has a value of "*ALL*" or "*FRIENDLYNAME*" then the device shall reset the value of the "*FriendlyNameStatus*" state variable `<friendlyName>` element value to the factory setting originally advertised in the device DDD.

If the *RestoreType* input parameter has a value of "*ALL*" or "*ICONLIST*" and the device supports the *SetFriendlyIconList()* action then the device shall reset its *FriendlyIconListStatus* state variable `<iconList>` elements to values indicating the factory setting along with any "*OPEN*" icon elements that the device allocates resources.

6.6.5.1 Arguments

Table 6-16: Arguments for *RestoreFriendlyInfo()*

Argument	Direction	relatedStateVariable
RestoreType	*IN*	*A_ARG_TYPE_RestoreType*

6.6.5.2 Device Requirements

If the device supports the *Security Feature* then this action is defined as a *Restrictable* action and the targeted device is allowed to restrict its invocation to specific control points which are identifiable or have specific *Roles*.

6.6.5.3 Dependency on State (if any)

None.

6.6.5.4 Effect on State (if any)

Successful invocation of this action will change the value of the *FriendlyNameStatus* state variable and cause it to be evented. If the device is in a state that allows it to leave the network and re-advertise its new *friendly* values then it should do so, however, it shall wait at least 30 seconds after any invocation of any *GetFriendlyName()*, *GetFriendlyIconList()*, *SetFriendlyName()*, *SetFriendlyIconList()* or *RestoreFriendlyInfo()* actions before initiating the re-advertisement behavior; this is done to provide opportunities for control points to make more than one change to the DDD without a re-advertisement. The timing for leaving and re-advertising is implementation dependent but shall be confomant to ISO/IEC 29341-1.

Once the device re-advertises the value of its DDD `<friendlyName>` element it should be the same as the value of the *FriendlyNameStatus* state variable `<friendlyName>` element immediately prior to the device leaving the network. Once re-advertised the *FriendlyNameStatus* state variable `<friendlyName>` element should be in the *clean* state.

Once the device re-advertises the value of its DDD `<iconList>` elements they should reflect the same values as all *FriendlyIconListStatus* state variable `<icon>` elements immediately prior to the device leaving the network Once re-advertised, the *FriendlyIconListStatus* state variable should be in the *clean* state. Note that `<getToken>` and `<postToken>` and `status@icon` are not part of the the DDD.

6.6.5.5 Errors

Table 6-17: Error Codes for *RestoreFriendlyInfo()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
700		Reserved for future extensions.

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

7 Theory of Operation (Informative)

This service enables UPnP control points to discover devices that allow their *friendly* DDD - `<friendlyName>` and `<iconList>` - information to be updated.

Control points can get (read) the current values, both *pending* and *clean*, of those *friendly* DDD elements using the *GetFriendlyName()* and *GetFriendlyIconList()* actions. The *GetFriendlyIconList()* action can also expose resources for creating new icons. Control points can set (write) the *friendly* DDD - `<friendlyName>` and `<iconList>` - information using the *SetFriendlyName()* and *SetFriendlyIconList()* actions respectively. Special timing considerations, to prevent race conditions, are enforced. If supported, a control point can request that the device restore its factory settings using the *RestoreFriendlyInfo()* action.

7.1 Changing the device `<friendlyName>`

In this example[1] the control point wants to change the `<friendlyName>` value of a device it has discovered where the current value of the `<friendlyName>` elements is

[1] In the following examples some tabs, whitespaces, and carriage return/line-feeds have been added within element values for readability and are not intended for exact interpretation.

<friendlyName>XYZ MediaServer</friendlyName>. It invokes the *SetFriendlyName()* action as follows:

```
Request (SOAP):
SetFriendlyName(
   "My Super ACME MediaServer in the Den"
)
```

As a result, the device grants the control point request to update the DDD <friendlyName>, but in this case, because it is in the process of serving content, it does not immediately leave the network and then re-advertise the new device. Instead, it immediately updates the *FriendlyNameStatus* state variable from the following value (assuming there are no *pending* DDD updates, that is, the device is in the *clean* state with regards to the <friendlyName>)

```
<?xml version="1.0" encoding="UTF-8"?>
<FriendlyNameStatus
 xmlns="urn:schemas-upnp-org:fd:fns-events"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:fd:fns-events
      http://www.upnp.org/schemas/fd/fns-events.xsd">
 <friendlyName status="DDD">ACME MediaServer</friendlyName>
</FriendlyNameStatus>
```

to

```
<?xml version="1.0" encoding="UTF-8"?>
<FriendlyNameStatus
 xmlns="urn:schemas-upnp-org:fd:fns-events"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:fd:fns-events
   http://www.upnp.org/schemas/fd/fns-events.xsd">
   <friendlyName status="PENDING">My Super ACME MediaServer
    in the Den
   </friendlyName>
</FriendlyNameStatus>
```

and events the change. The device then waits at least 30 seconds to see if there are any additional invocations of the *GetFriendlyName()*, *GetFriendlyIconList()*, *SetFriendlyName()*, *SetFriendlyIconList()* or *RestoreFriendlyInfo()* action to see if a re-advertisement is OK. Since it was in the process of serving some content it waits until that session is complete.

7.2 Changing the device <iconList>

In this example[1], the device indicates in its *FriendlyIconListStatus* state variable all of its <iconList> information including two existing icons already advertised in the DDD and two open icon slots for posting new icon binaries – one for JPEG and one for PNG.

```
<?xml version="1.0" encoding="UTF-8"?>
<FriendlyIconListStatus
 xmlns="urn:schemas-upnp-org:fd:fis-events"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:fd:fis-events
```

[1] In the following examples some tabs, whitespaces, and carriage return/line-feeds have been added within element values for readability and are not intended for exact interpretation.


```
   http://www.upnp.org/schemas/fd/fis-events.xsd">
   <iconList>
      <icon status="DDD">
         <mimetype>image/png</mimetype>
         <height>80</height>
         <width>80</width>
         <depth>24</depth>
         <url>icons/zbot720smallIconBinary.png</url>
      </icon>
      <icon status="DDD">
         <mimetype>image/png</mimetype>
         <height>320</height>
         <width>320</width>
         <depth>24</depth>
         <url>icons/zbot720mediumIconBinary.png</url>
      </icon>
      <icon status="OPEN" maxBytes="10000000">
         <mimetype>image/jpg</mimetype>
         <getToken>get-003-jpg</getToken>
         <postToken postUri="http://192.168.1.5[1]/icons/jpg?post">
            post-003-jpg
         </postToken>
      </icon>
      <icon status="OPEN" maxBytes="10000000">
         <mimetype>image/png</mimetype>
         <getToken>get-004-png</getToken>
         <postToken postUri="http://192.168.1.5/icons/png?post">
            post-004-png
         </postToken>
      </icon>
   </iconList>
</FriendlyIconListStatus>
```

The control point invokes a *SetFriendlyIconList()* action targeting a replacement[2] of the "medium" icon with a new "bigger" icon binary located somewhere in the network at URI `http://192.168.1.15:80/iconlocation/zbot720biggerIconBinary.png`. Since the device supports both HTTP-POST and HTTP-GET the control point, uses the `<getToken>` element (not the `<postToken>` element) in the *TargetIconToken* input argument to indicate that the device shall use an HTTP-GET to create the new icon binary. The action is as follows:

Request (SOAP):

```
SetFriendlyIconList(
   "CREATE",
   "get-004-png",
   "http://192.168.1.15:80/iconlocation/zbot720biggerIconBinary.png"
)
```

Before returning an action response the device begins execution of an HTTP-GET to *TargetIconURI*
`http://192.168.1.15:80/iconslocation/zbot720biggerIconBinary.png`.

Suppose, in the mean time another control point invokes a *SetFriendlyIconList()* action targeting an update of the "small" icon with an icon binary that it plans to post as follows:

Request (SOAP):

```
SetFriendlyIconList(
   "CREATE",
```

[1] The IP address may be explicit or understood to be relative to the device.
[2] The replacement will be accomplished by first creating a new icon and then deleting the existing.

