INTERNATIONAL STANDARD

ISO/IEC 29341-29-10

First edition
2017-09

Information technology — UPnP Device Architecture —

Part 29-10: Multiscreen device control protocol — Level 2 — Application management service

Technologies de l'information — Architecture de dispositif UPnP —

Partie 29-10: Protocole de contrôle de dispositif multi-écran — Niveau 2 — Service de gestion des applications

Reference number
ISO/IEC 29341-29-10:2017(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

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 29341-29-10 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 document specifies the characteristics of the UPnP networked service named *ApplicationManagement*, version *2*. This service definition is compliant with UPnP Device Architecture 1.0 [1].

This service type enables to manage applications and the communications between applications providing various time-sensitive and interactive services including implementation-specific applications among various display devices, that is, Screen Devices [3] and Screen Control Points..

Screen Devices shall implement this service [3], but this service is allowed to be implemented for any UPnP devices as an add-on service.

2 Normative references

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

[1] – *UPnP Device Architecture, version 1.0*, UPnP Forum, October 15, 2008.
Available at: http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0-20081015.pdf.
Latest version available at: http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0.pdf.

[2] – *Multi-Screen Architecture:1*, UPnP Forum, September 30, 2014.
Available at: http://www.upnp.org/specs/ms/UPnP-ms-MultiScreenArchitecture-v1-20140930.pdf.
Latest version available at: http://www.upnp.org/specs/ms/UPnP-ms-MultiScreenArchitecture-v1.pdf.

[3] – *ScreenDevice:2*, UPnP Forum, September 30, 2014.
Available at: http://www.upnp.org/specs/ms/UPnP-ms-ScreenDevice-v2-Device-20140930.pdf.
Latest version available at: http://www.upnp.org/specs/ms/UPnP-ms-ScreenDevice-v2-Device.pdf.

[4] – *IETF RFC 3986, Uniform Resource Identifiers (URI): Generic Syntax*, January 2005.
Available at: http://www.ietf.org/rfc/rfc3986.txt.

[5] – *IETF RFC 1738, Uniform Resource Locators (URL)*, December 1994.
Available at: http://www.ietf.org/rfc/rfc1738.txt.

[6] – *IETF RFC 6455, The WebSocket Protocol*, December 2011.
Available at: http://www.ietf.org/rfc/rfc6455.txt.

[7] – *IETF RFC-6120, Extensible Messaging and Presence Protocol XMPP: Core*, March 2011.
Available at http://tools.ietf.org/html/rfc6120

[8] – *MediaRenderer:3*, UPnP Forum, March 31, 2013.
Available at: http://www.upnp.org/specs/av/UPnP-av-MediaRenderer-v3-Device-20130331.pdf.
Latest version available at: http://www.upnp.org/specs/av/UPnP-AV-MediaRenderer-v3-Device.pdf.

[9] – *DeviceProtection:1*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/specs/gw/UPnP-gw-DeviceProtection-v1-Service-20110224.pdf.
Latest version available at: http://www.upnp.org/specs/gw/UPnP-gw-DeviceProtection-v1-Service.pdf.

[10] – *XML Schema for FeatureList XML Structures*, UPnP Forum, September 30, 2014.
Available at: http://www.upnp.org/ schemas/ms/FeatureList-v1-20140930.xsd.

Latest version available at: http://www.upnp.org/ schemas/ms/FeatureList.xsd.

[11] – *XML Schema for AppInfoList XML Structures*, UPnP Forum, September 30, 2014.
Available at: http://www.upnp.org/ schemas/ms/AppInfoList-v2-20140930.xsd.
Latest version available at: http://www.upnp.org/ schemas/ms/AppInfoList.xsd.

3 Terms, definitions, symbols and abbreviations

For the purposes of this document, the terms and definitions given in the UPnP Device Architecture [1], the Multi-Screen Architecture:1 [2] and the following apply.

3.1 Terms specific to ApplicationManagement

3.1.1 *AMS features*

A set of extended functionalities for the ApplicationManagement service with additional requirements beyond the general ApplicationManagement service mechanisms (see subclause 5.2.1).

3.1.2 *SECURITY feature*

One of *AMS features* and an extension of the DeviceProtection service [9] to the actions (*Action Level Access*) of the ApplicationManagement service (see subclause 5.2.2).

3.1.3 *AM Roles*

Access level of a *Control Point* or *User Identity* to authorize a specific set of the ApplicationManagement actions (see subclauses 5.2.2.1 and 5.2.2.2).

3.1.4 Application

A software program designed to help people perform an activity.

3.1.5 Native Application

A type of application running directly on an OS platform. Typically, it needs to be installed before it can be started.

3.1.6 Web Application

A type of application typically written with web-native languages such as HTML, JavaScript and so on. It runs directly in a web browser. Typically, it does not need to be installed before it can be started.

4 Notations and Conventions

See the Multi-Screen Architecture:1 [2].

5 Service Modelling Definitions

5.1 Service Type

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

urn:schemas-upnp-org:service:*ApplicationManagement:2*

5.2 Key Concepts

5.2.1 *AMS features*

This subclause defines a set of extended functionalities for the ApplicationManagement service, called *AMS features*. These features have additional requirements beyond the general ApplicationManagement service mechanisms to ensure interoperability. When an implementation supports a specific *AMS feature*, it shall support that feature according to the rules in this subclause.

Each *AMS feature* shall have an integer version number. Later versions – indicated by a larger version number – shall support the full functionality of all earlier, lower-numbered

versions in the same way as the earlier version (that is, shall be backward compatible). See subclause 5.3.1 and the schema [10] for more details.

Table 1 — *AMS features*

Feature Name	Version	Description	Requirements
Default (no feature)		Default functionalities	*GetAppInfoByID()* *GetSupportedTargetFields()* *GetAppIDList()* and the related state variables
START	1	Application-starting related functionalities	*StartAppByID()* *StartAppByURL()* *StopApp()* *GetRunningAppList()* *GetRunningStatus()* and the related state variables
INSTALL	1	Application-installing related functionalities	*InstallAppByID()* *InstallAppByURI()* *UninstallApp()* *GetInstallationStatus()* and the related state variables
CONNECT	1	Application-connecting related functionalities	*GetAppConnectionInfo()* *ConnectApptoApp()* *DisconnectApptoApp()* *GetCurrentConnectionInfo()* and the related state variables
SECURITY	1	Action Level Access	See subclause 5.2.2.

If an ApplicationManagement service implementation supports an *AMS feature*, it shall support the associated requirements described in Table 1. In addition, each action is allowed to be supported only if its associated *AMS feature* is supported. I.e. all the actions associated with an *AMS feature* shall be supported simultaneously, or none of them is allowed to be supported.

An ApplicationManagement service implementation is allowed to support multiple *AMS features*. The Default functionalities described in Table 1 shall be supported by default without association with any *AMS feature*. The *START feature* shall be supported by the ApplicationManagement service version 2.

5.2.2 *SECURITY feature*

The *SECURITY feature* is an extension of the DeviceProtection service [9] to the actions (*Action Level Access*) of the ApplicationManagement service. The *SECURITY feature* is only allowed to be supported on a device which also implements the DeviceProtection service [9], and not allowed otherwise.

By defining *Action Level Access* based on the *Roles* defined by the DeviceProtection service [9] and the ApplicationManagement service, a ScreenDevice is able to restrict access from unidentified Control Points or Users, and to differentiate access levels for identified Control Points or Users with different *Roles*. Additionally, an implementation may define other vendor *Roles* with other *Action Level Access*.

If a Control Point has at least one *Role* that is not restricted from invoking a specific action, then it is said to have *Action Level Access*. Otherwise, the ApplicationManagement service implementation shall issue the error code 606 (see the UPnP Device Architecture [1]) in response to the action invocation.

Table 2 — Error Codes for *Action Level Access*

errorCode	errorDescription	Description
400-499	TBD	See clause 3 in the UPnP Device Architecture [1].
500-599	TBD	See clause 3 in the UPnP Device Architecture [1].
606	Action not authorized	Action not authorized: The Control Point does not have privileges to invoke this action

5.2.2.1 AM (ApplicationManagement) *Roles*

The following Table 3 lists pre-defined *AM Roles* for the *SECURITY feature.* These *Roles* shall be supported when the *SECURITY feature* is implemented. This list of pre-defined *Roles* may be extended by the implementer with additional *vendor-defined Roles*.

Table 3 — ApplicationManagement *Roles*

Role Name	R/A
Public	*CR*
Basic	*CR*
AM_SuperUser	*CR*
Admin	*CR*
Conditionally required if the *SECURITY feature* is implemented, and not allowed otherwise.	

The *Public Role* is defined in the DeviceProtection service [9]. This role is assigned limited read-related action permissions including actions revealing non-personalized information among the ApplicationManagement service state variables to control points (see Table 4 for details). This is the default DeviceProtection service *Role* and therefore default *AM Role*.

The *Basic Role* is defined in the DeviceProtection service [9]. This *Role* is assigned full read-related action permissions including actions revealing information among the ApplicationManagement service state variables to control points. This *Role* is also assigned limited write-related action permissions including actions changing non-installation-related information among the ApplicationManagement service state variables (see Table 4 for details).

The *AM_SuperUser Role* is defined in the ApplicationManagement service. This *Role* is assigned full *Action Level Access* (see Table 4 for details). Assignment of the *AM_SuperUser Role* to an unrecognized *User* or *Control Point Identity* is not allowed.

The *Admin Role* is defined by the DeviceProtection service [9]. The *Admin Role* has no effect with regards to the ApplicationManagement actions. However, a Control Point with the *Admin Role* is allowed to add the *Roles* to any *User* or *Control Point Identity* enabling this *Identity* to have proper permissions for all ApplicationManagement service actions.

5.2.2.2 *Restrictable*/*Non-Restrictable* Actions and *Action Level Access*

ApplicationManagement actions are defined as *Restrictable* or *Non-Restrictable* (see Table 4) only when the *SECURITY feature* is supported.

The Table 4 shows ApplicationManagement actions accessible to a *User* or *Control Point Identity* assigned each of the *Roles*. A *User* or *Control Point Identity* possessing more than one of these *Roles* would be allowed to access to any action permitted by any of the assigned *Roles*. A *YES* value indicates that *Action Level Access* shall be granted by the corresponding *Role*, while a *NO* value indicates that *Action Level Access* shall not granted by this Role. Note that a *NO* value does not explicitly prohibit *Action Level Access*. That is, another *Role* that a *User* or *Control Point Identity* possesses may permit *Action Level Access*.

Table 4 — Action to *Role* Permission Mapping

Action Name	Category	Role		
		Public	***Basic***	***AM Super User***
GetFeatureList()	*Non-Restrictable*	*YES*	*YES*	*YES*
GetAppInfoByIDs()	*Restrictable*	*NO*	*YES*	*YES*
GetSupportedTargetFields()	*Non-Restrictable*	*YES*	*YES*	*YES*
GetAppIDList()	*Restrictable*	*NO*	*YES*	*YES*
GetRunningAppList()	*Restrictable*	*NO*	*YES*	*YES*
GetRunningStatus()	*Restrictable*	*NO*	*YES*	*YES*
StartAppByID()	*Restrictable*	*NO*	*YES*	*YES*
StartAppByURI()	*Restrictable*	*NO*	*YES*	*YES*
StopApp()	*Restrictable*	*NO*	*YES*	*YES*
InstallAppByID()	*Restrictable*	*NO*	*NO*	*YES*
InstallAppByURL()	*Restrictable*	*NO*	*NO*	*YES*
UninstallApp()	*Restrictable*	*NO*	*NO*	*YES*
GetInstallationStatus()	*Restrictable*	*NO*	*YES*	*YES*
GetAppConnectionInfo()	*Restrictable*	*NO*	*YES*	*YES*
ConnectApptoApp()	*Restrictable*	*NO*	*YES*	*YES*
DisconnectApptoApp()	*Restrictable*	*NO*	*YES*	*YES*
GetCurrentConnectionInfo()	*Restrictable*	*NO*	*YES*	*YES*

5.3 State Variables

Note: For a first-time reader, it might be more helpful to read the action definitions before reading the state variable definitions.

Table 5 —State variables

Variable Name	R/A [a]	Data Type	Allowed Value	Default Value	Eng. Units
FeatureList	*R*	**string**	CSV(**string**) See subclause 5.3.1		
A_ARG_TYPE_AppIDs	*R*	**string**	CSV(**string**) See subclause 5.3.2		
AppInfoList	*R*	**string**	*AppInfoList XML Document* See subclause 5.3.3		
A_ARG_TYPE_AppInfo	*R*	**string**	*XML fragment of AppInfoList XML Document* See subclause 5.3.4		
SupportedTargetFields	*R*	**string**	CSV(**string**) See subclause 5.3.5		
A_ARG_TYPE_Target	*R*	**string**	See subclause 5.3.6		
A_ARG_TYPE_TargetFields	*R*	**string**	CSV(**string**) See subclause 5.3.7		
RunningAppList	*CR* c	**string**	CSV(**string**) See subclause 5.3.8		
TransitioningApps	*CR* c	**string**	CSV(**string**) See subclause 5.3.9		
A_ARG_TYPE_URI	*CR* c	**string**	See subclause 5.3.10		
A_ARG_TYPE_Parameter	*CR* c	**string**	See subclause 5.3.11		
A_ARG_TYPE_ConnectionIDs	*CR* b	**string**	CSV(**string**) See subclause 5.3.12		
Non-standard state variables implemented by a UPnP vendor go here	*X*	*TBD*	*TBD*	*TBD*	*TBD*

a For a device this column indicates whether the state variable 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*, *A-D*).

b *CR* = conditionally required. See referenced subclause for implementation requirements.

c *CR* = conditionally required. In fact required since the condition is required for this specification. See referenced subclause for implementation requirements.

5.3.1 *FeatureList*

This conditionally required state variable shall be supported if any of the *AMS features* is supported, and allowed otherwise. The state variable enumerates the *AMS features* supported by this ApplicationManagement service (see subclause 5.2.1 for details). The following is the XML template for the *FeatureList* state variable. See the schema in [10] for more details on the structure.

```
<?xml version="1.0" encoding="UTF-8"?>
<FeatureList
  xmlns="urn:schemas-upnp-org:ms:FeatureList"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    urn:schemas-upnp-org:ms:FeatureList
       http://www.upnp.org/schemas/ms/FeatureList.xsd">
    <feature name="Feature Name" version="Feature Version"></feature>
</FeatureList>
```

5.3.2 *A_ARG_TYPE_AppIDs*

This required state variable provides type information for the various *application@id*-related arguments in various actions. This state variable is a CSV list of the *application@id* values defined in the *AppInfoList* state variable (see subclause 5.3.3).

5.3.3 *AppInfoList*

This required state variable contains overall information of applications which a Screen Device is having for *multi-screen services*. The following is the XML template for the *AppInfoList* state variable. See the schema in [11] for more details on the structure.

```
<?xml version="1.0" encoding="UTF-8"?>
<AppInfoList
 xmlns="urn:schemas-upnp-org:ms:AppInfoList"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:ms:AppInfoList
      http://www.upnp.org/schemas/ms/AppInfoList.xsd">
   <application id="Unique Identifier of the Application">
      <marketAppID market="Market Name Providing the Application"
      version="Application Version">
         Market-assigned ID of Application</marketAppID>
      <friendlyName language="Language of friendlyName">
         Short User-friendly Title</friendlyName>
      <alternativeID org="Organization Name">Standard Organization-assigned ID of
         Application</alternativeID>
      <function org="Organization Name">Standard Organization-assigned ID of
         functionality implemented by Application</function>
      <runningStatus>Activation Status of Application</runningStatus>
      <startURI deviceType="Applicable Device Type">URI for Activation of the
         Application</startURI>
      <installationStatus>Installation Status of Application</installationStatus>
      <downloadingProgress>Percentage of the Application that has been
         downloaded</downloadingProgress>
      <installationProgress>Percentage of the Application that has been
         installed</installationProgress>
      <installationURI deviceType="Applicable Device Type">URI for Installation of
         the Application</installationURI>
      <usagePolicy>Restriction Information</usagePolicy>
      <apptoAppInfo>
         <matchingProtocolName>User-friendly protocol name</matchingProtocolName>
         <protocol required="1 or 0">Protocol name</protocol>
         <connectionAddress>Information needed to establish a connection to the
            Application</connectionAddress>
      </apptoAppInfo>
      <iconList>
         <icon>
            <mimetype>MIME type of Application image</mimetype>
            <width>Width of Application image, in pixels</width>
            <height>Height of Application image, in pixels</height>
            <depth>Bit-depth of Application image, in bits</depth>
            <url>HTTP URL for downloading Application image</url>
         </icon>
      </iconList>
   </application>
</AppInfoList>
```

<xml>
Allowed. Case sensitive.

<AppInfoList>
Required. <XML>. Shall include a namespace declaration for the XML Schema for AppInfoList XML Structures [11] (“urn:schemas-upnp-org:ms:AppInfoList”). Shall include zero or more of the following elements.

<application>
Required. <XML>. Shall appear once for each application. Contains the following attributes and sub-elements:

@id
Required. xsd:string. Provide a unique identity (i.e., UUID. See the UPnP Device Architecture [1].) for the application within the ApplicationManagement service.

<marketAppID>
Allowed. xsd:string. Provides the identifier of an application which is assigned by an application market. Contains the following attributes:

@market
Required. xsd:string. Indicates the identification of the digital distribution platform which the application is provided by.

@version
Required. xsd:string. Provides a version of the application. This is a literal string that denotes a version. String comparison will be done to determine if a version is higher. For example, 123.345.456 is higher than 123.245.999, BetaVersion_1 is higher than Alphaversion_2.

<friendlyName>
Required. xsd:string. Provides a short description (e.g., title) of the application for end user. Shall appear once for each different friendly name. This value can be used for Screen Control Point(s) to search an appropriate application when the <*marketAppID*> element is not correctly interpreted. May be localized (see *@language*).

@language
Allowed. xsd:string. Indicates the language of the <*friendlyName*> element. See RFC 1766 language tag(s).

<alternativeID>
Allowed. xsd:string. Provides an identifier of the application used for standard organizations. Shall appear once for each different alternative ID.

@org
Required. xsd:string. Provides the domain name of the organization using the <*alternativeID*> value.

<function>
Allowed. xsd:string. Provides an identifier of the functionality implemented by the application. Shall appear once for each different functionality identifier. See Table for details.

@org
Required. xsd:string. Provides the domain name of the organization that has defined the <*function*> value.

<runningStatus>
Conditionally required. Shall be supported if the *START feature* is supported, and allowed otherwise. xsd:string. Indicates the activation status of the application on the Screen Device. The allowed values are "*Inactive*", "*Transitioning*", "*Transitioning_Pending_Input*", "*Running*", and "*Unknown*".

<startURI>
Allowed. xsd:anyURI. Contains a URI which Screen Control Point(s) can access in order to start the application. Shall appear once for each different device type.

@deviceType
Required. xsd:string. Indicates the device type which the application is applicable to. The allowed values are "*Both*", "*Main_Screen_Device*", and "*Companion_Screen_Device*".

<installationStatus>
Conditionally required. xsd:string. Shall be supported if the *INSTALL feature* is supported, and allowed otherwise. Indicates the installation status of the application specified by the <*appID*> element on the MainScreen Device. The allowed values are "*Not_Downloaded*", "*Downloading*", "*Downloading_Pending_Input*", "*Not_Installed*", "*Installing*", "*Installing_Pending_Input*", "*Installed*", and "*Unknown*". The values of "*Not_Downloaded*", "*Downloading*" and "*Downloading_Pending_Input*" are applicable only to applications which need to be downloaded for installation. For those applications, the value of "*Not_Installed*" indicates a status that the application has been downloaded but not been installed yet. For applications which do not need installation to be activated, this element shall have the value of "*Installed*".

<downloadingProgress>
Allowed. xsd:unsignedShort.. Allowed to appear only if the <*installationStatus*> element has the "*Downloading*" value, and not allowed otherwise. Indicates a percentage value representing the progress of the download. Allowed to have an integer value from "*0*" to "*100*". "*0*" means no byte downloaded yet, and "*100*" means fully downloaded. The determination of this value is implementation specific, for example based on the number of bytes processed.

<installationProgress>
Allowed. xsd:unsignedShort.. Allowed to appear only if the <*installationStatus*> element has the "*Installing*" value, and not allowed otherwise. Indicates a percentage value representing the progress of the Installation. Allowed to have an integer value from "*0*" to "*100*". "*0*" means not installed at all and "*100*" means fully installed. The determination of this value is implementation specific, for example based on the number of bytes processed.

<installationURI>
Allowed. xsd:anyURI. Contains a URI which Screen Control Point(s) can access in order to install the application. Shall appear once for each different device type.

@deviceType
Required. xsd:string. Indicates the device type which the application specified by <*appID*> element is applicable to. The allowed values are "*Both*", "*Main_Screen_Device*", and "*Companion_Screen_Device*".

<usagePolicy>
Allowed. xsd:string. Indicates permission related information for using the application. The allowed values are "*No_Restriction*", "*Purchase_Required*", "*Trial_Only*", "*Parental_Consent_Required*", "*Sign-in_Required*" and "*Unknown*".

<apptoAppInfo>
Conditionally required. <XML>. Shall be supported if the *CONNECT feature* is supported, and allowed otherwise. Shall be supported at least when the <*runningStatus*> is set to "*Running*". Provides the information to make an app-to-app connection to the application. Shall appear once for each different connection information. Contains all of the following attribute and sub-element:

<matchingProtocolName>
Required. xsd:string. Provides a *vendor/organization-defined* protocol name used for the App-to-App communication over a transport layer specified by the <*protocol*> element. This contains the ICANN assigned domain name owned by the vendor/organization followed by underscore "_" and the version number of the application's communication protocol. This field can be used to find a communication-compatible application(s).

<protocol>
Required. xsd:string. Provides the protocol of the transport layer for the App-to-App communication. The allowed values are "*HTTP*", "*Websocket*", "*XMPP*", "*UPnP*" and v*endor-defined*.

@required
Required. xsd:boolean. Indicates whether the Screen Control Point is required to use the communication channel described in the <*protocol*> to communicate to the running application. "*1*" means that the <*protocol*> is required and "*0*" means that it is not required.

<connectionAddress>
Allowed. xsd:string. Provides the access information for the App-to-App communication. The syntax of this element varies depending on the value of the <*protocol*>. See Table for details. Note that when the <*runningStatus*> is not set to "*Running*", this element is allowed to be omitted. The omission of this element is implementation dependent.

<iconList>
Allowed. <XML>. Contains the following subelements:

<icon>
Recommended. <XML>. Icon to depict the application in a Screen Control Point UI. Icon sizes to support are vendor-specific. Shall appear once for each different icon. Contains the following sub elements:

<mimetype>
Required. xsd:string. Icon's MIME type (cf. RFC 2045, 2046, and 2387). Single MIME image type. At least one icon should be of type "image/png" (Portable Network Graphics, see IETF RFC 2083).

<width>
Required. xsd:unsignedInt. Horizontal dimension of icon in pixels.

<height>
Required. xsd:unsignedInt. Vertical dimension of icon in pixels.

<depth>
Required. xsd:unsignedInt. Number of color bits per pixel.

<url>
Required. xsd:anyURI. Pointer to icon image. Retrieved via HTTP. Shall be relative to the URL at which the device description is located in accordance with section 5 of RFC 3986. Specified by UPnP vendor.

Table 6 — Allowed values for *function* element

@org value	*function* value	Reference
upnp.org	urn:schemas-upnp-org:device:deviceType:v	[1]
upnp.org	urn:schemas-upnp-org:service:serviceType:v	[1]
vendor-defined	*vendor-defined*	

Table 7 — Allowed values for *connectionAddress* element

protocol value	*connectionAddress* value	Reference
HTTP	An absolute http or https URI	[4], [5]
Websocket	WebSocket URI	[6]
UPnP	uuid:device-uuid	[1]
vendor-defined	*vendor-defined*	

The following is an example where the *AppInfoList* state variable contains information about two applications, "SimpleMediaPlayer" and "AdvancedMediaPlayer".

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<AppInfoList
 xmlns="urn:schemas-upnp-org:ms:AppInfoList"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:ms:AppInfoList
      http://www.upnp.org/schemas/ms/AppInfoList.xsd">
   <application id="F58E1D3B-859A-40EC-928E-A5889EF0B458">
      <marketAppID market="MyAppStore" version="1">
         SimpleMediaPlayer/OSZ/64bit/v1</marketAppID>
      <friendlyName>Simple Media Player</friendlyName>
      <runningStatus>Inactive</runningStatus>
      <usagePolicy>No_Restriction</usagePolicy>
   </application>
   <application id="CB0D5D97-29F9-488B-AE6B-7D6B4136112B">
      <marketAppID market="MyAppStore" version="1">
         AdvancedMediaPlayer/OSZ/64bit/v1</marketAppID>
      <friendlyName>Advanced Media Player</friendlyName>
      <runningStatus>Running</runningStatus>
      <usagePolicy>No_Restriction</usagePolicy>
      <apptoAppInfo>
         <matchingProtocolName>HTTP_UPnP.org_v1</matchingProtocolName>
         <protocol required="1">HTTP</protocol>
         <connectionAddress>
            http://192.168.0.50:34567/apps/AdvancedMediaPlayer/connect
         </connectionAddress>
      </apptoAppInfo>
   </application>
</AppInfoList>
```

The following is an example where the *AppInfoList* state variable contains information about an application called, “MediaRendererApp”. This app implements a UPnP MediaRenderer:3 device [8].

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<AppInfoList
 xmlns="urn:schemas-upnp-org:ms:AppInfoList"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:ms:AppInfoList
      http://www.upnp.org/schemas/ms/AppInfoList.xsd">
   <application id="5E0E4EC1-6CC4-4D12-9995-7F996B709726">
      <marketAppID market="MyAppStore" version="1">
         MediaRendererApp/OSZ/64bit/v1</marketAppID>
      <friendlyName>Media Renderer</friendlyName>
      <function org="upnp.org">
         urn:schemas-upnp-org:device:MediaRenderer:3
      </function>
      <runningStatus>Running</runningStatus>
      <usagePolicy>No_Restriction</usagePolicy>
      <apptoAppInfo>
         <matchingProtocolName>UPnP_3</matchingProtocolName>
         <protocol required="1">XMPP</protocol>
         <connectionAddress>
            uuid:18306773-E98C-4309-A5FB-EEB38C2A1F75
         </connectionAddress>
      </apptoAppInfo>
   </application>
</AppInfoList>
```

5.3.4 *A_ARG_TYPE_AppInfo*

This required state variable provides type information for arguments in various actions. The state variable shall be an XML fragment of the XML document for the *AppInfoList* state

variable (see subclause 5.3.3). It shall contain zero or more <*application*> element(s), its (their) attributes and sub-elements which depend on the invoked actions.

5.3.5 *SupportedTargetFields*

This required state variable provides an unordered CSV list of the searchable fields by the *GetAppIDList()* action (see subclause 5.5.4), that is, elements or attributes of the *AppInfoList* state variable of the Screen Device. The value which each component of the CSV list is allowed to have is any element name without its parent element name, or any attribute name following its element name without its parent element name. For example, *usagePolicy*, *alternativeID@org* and so on. This state variable shall contain the required fields in Table 8. The required fields can be expanded in future versions of the specification.

Table 8 — Required fields for *SupportedTargetFields*

Value	Parameter in *AppInfoList*	R/A
friendlyName	*application::friendlyName*	*R*
matchingProtocolName	*application::apptoAppInfo:: matchingProtocolName*	*R*

5.3.6 *A_ARG_TYPE_Target*

This required state variable provides type information the *Target* input argument in the *GetAppIDList()* action (see subclause 5.5.4)

5.3.7 *A_ARG_TYPE_TargetFields*

This required state variable provides type information for the *TargetFields* input argument in the *GetAppIDList()* action (see subclause 5.5.4). This state variable is an unordered CSV list of the values in the CSV list of the *SupportedTargetFields* state variable.

5.3.8 *RunningAppList*

This conditionally required state variable shall be supported if the *START feature* is supported (see subclause 5.2.1 and 5.3.1), and allowed otherwise. This state variable provides a list of running applications for the *RunningAppList* output argument in the *GetRunningAppList()* action and eventing. The state variable is a CSV list of the *@id* values of the <*application*> elements of which their <*runningStatus*> value are set to "*Running*" in the *AppInfoList* state variable (see subclause 5.3.3). This state variable shall have an empty string when there are no applications with the <*runningStatus*> element set to "*Running*".

5.3.9 *TransitioningApps*

This conditionally required state variable shall be supported if the *START* or *INSTALL feature* is supported (see subclauses 5.2.1 and 5.3.1), and allowed otherwise. The state variable is a CSV list of a pair of the *@id* (or *StartURI* or *InstallURI* for those not assigned an *@id*) and either the <*runningStatus*> or <*installStatus*> value of the <*application*> elements under transitioning status. The transitioning status means that any <*runningStatus*> value is neither "*Inactive*", "*Running*" nor "*Unknown*", or any <*installStatus*> value is neither "*Not_Downloaded*", "*Not_Installed*", "*Installed*" nor "*Unknown*". This state variable shall have an empty string when there are no applications under transitioning status.

5.3.10 *A_ARG_TYPE_URI*

This conditionally required state variable shall be supported if the *START feature* or the *INSTALL feature* is supported (see subclauses 5.2.1 and 5.3.1), and allowed otherwise. This state variable provides type information for the *StartURI* and *InstallationURI* input arguments in the *StartAppByURI()* and *InstallAppByURI()* actions (see subclauses 5.5.8 and 5.5.11). This state variable shall be properly escaped as described in [4]. In addition, it shall be escaped according to the requirements in [5].

5.3.11 *A_ARG_TYPE_Parameters*

This conditionally required state variable shall be supported if the *START feature* or the *INSTALL feature* is supported (see subclauses 5.2.1 and 5.3.1), and allowed otherwise. This state variable provides type information for the *StartParameters* and *InstallParameters* input

arguments in various actions (see subclauses 5.5.7, 5.5.8, 5.5.10 and 5.5.11). The arguments are used for the according actions to be successfully accepted, and the proper values are application-specific.

5.3.12 *A_ARG_TYPE_ConnectionIDs*

This conditionally required state variable shall be supported if the *CONNECT feature* is supported (see subclauses 5.2.1 and 5.3.1), and allowed otherwise. This state variable provides type information for the arguments in various actions. This state variable is a CSV list of unique identifiers of app-to-app connections (i.e., UUID. See the UPnP Device Architecture [1].) currently supported by a Screen Device within the ApplicationManagement service.

5.4 Eventing and Moderation

Table 9 — Event moderation

Variable Name	Evented	Moderated Event	Min Event Interval [a] (seconds)	Logical Combination	Min Delta per Event [b]
FeatureList	*NO*	*NO*			
A_ARG_TYPE_AppIDs	*NO*	*NO*			
AppInfoList	*NO*	*NO*			
A_ARG_TYPE_AppInfo	*NO*	*NO*			
SupportedTargetFields	*NO*	*NO*			
A_ARG_TYPE_Target	*NO*	*NO*			
A_ARG_TYPE_TargetFields	*NO*	*NO*			
RunningAppList	*YES*	*YES*	0.2		
TransitioningApps	*YES*	*YES*	0.2		
A_ARG_TYPE_URI	*NO*	*NO*			
A_ARG_TYPE_Parameters	*NO*	*NO*			
A_ARG_TYPE_ConnectionIDs	*NO*	*NO*			
Non-standard state variables implemented by a UPnP vendor go here	*TBD*	*TBD*	*TBD*	*TBD*	*TBD*

a Max event rate is determined by N, where $Rate = 1/N$, where N is the Min Event Interval in seconds.

b (N) * (allowedValueRange Step)

5.5 Actions

The following tables and subclauses define the various ApplicationManagement service actions.

Except where noted, if an invoked action returns an error, the state of the device will be unaffected.

Table 10 — Actions

Name	R/A [a]	Control Point R/A [b]
GetFeatureList()	*CR* c	*A*
GetAppInfoByIDs()	*R*	*R*
GetSupportedTargetFields()	*R*	*A*
GetAppIDList()	*R*	*R*
GetRunningAppList()	*CR* d	*A*
GetRunningStatus()	*CR* d	*A*
StartAppByID()	*CR* d	*A*
StartAppByURI()	*CR* d	*A*
StopApp()	*CR* d	*A*
InstallAppByID()	*CR* c	*A*
InstallAppByURL()	*CR* c	*A*
UninstallApp()	*CR* c	*A*
GetInstallationStatus()	*CR* c	*A*
GetAppConnectionInfo()	*CR* c	*A*
ConnectApptoApp()	*CR* c	*A*
DisconnectApptoApp()	*CR* c	*A*
GetCurrentConnectionInfo()	*CR* c	*A*
Non-standard actions implemented by an UPnP vendor go here.	*X*	*X*

a 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*, *A-D*).

b 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*, *A-D*).

c *CR* = conditionally required. See referenced subclause for implementation requirements.

d *CR* = conditionally required. In fact required since the condition is required for this specification. See referenced subclause for implementation requirements.

Note that non-standard actions shall be implemented in such a way that they do not interfere with the basic operation of the ApplicationManagement service, that is: these actions shall be allowed and do not need to be invoked for the ApplicationManagement service to operate normally.

5.5.1 *GetFeatureList()*

This conditionally required action shall be supported if the *FeatureList* state variable is supported, and not allowed otherwise. The action enables a Screen Control Point to retrieve the *FeatureList* state variable (see subclauses 5.2.1 and 5.3.1).

5.5.1.1 Arguments

Table 11 — Arguments for *GetFeatureList()*

Argument	Direction	Related State Variable
FeatureList	*OUT*	*FeatureList*

5.5.1.2 Dependency on State

None.

5.5.1.3 Effect on State

None.

5.5.1.4 Errors

Table 12 — Error Codes for *GetFeatureList()*

errorCode	errorDescription	Description
400-499	TBD	See clause 3 in the UPnP Device Architecture [1].
500-599	TBD	See clause 3 in the UPnP Device Architecture [1].
600-699	TBD	See clause 3 in the UPnP Device Architecture [1].

5.5.2 *GetAppInfoByIDs()*

This required action enables a Screen Control Point to retrieve information of applications which are specified by the *AppIDs* input argument.

5.5.2.1 Arguments

- *AppIDs*: Specifies applications to retrieve their information. See subclause 5.3.2. The special value "*" means everything, i.e., the whole *AppInfoList* state variable will be retrieved.
- *AppInfo*: an XML fragment of the *AppInfoList* state variable (see subclauses 5.3.3 and 5.3.4). It shall contain the <*application*> elements (of which their *@id* values are identical to the values of the *AppIDs* input argument), and all their supported attributes and sub-elements. If any value of the *AppIDs* input argument is not valid, it shall return either error code 701 or respond with an *AppInfo* output argument containing <*application*> elements corresponding only to the valid values of the *AppIDs* input argument. The number of <*application*> elements of the *AppInfo* output argument shall be less than or equal to the number of *application@id*s included in the *AppIDs* input argument.

Table 13 — Arguments for *GetAppInfoByIDs()*

Argument	Direction	Related State Variable
AppIDs	*IN*	*A_ARG_TYPE_AppIDs*
AppInfo	*OUT*	*A_ARG_TYPE_AppInfo*

5.5.2.2 Dependency on State

None.

5.5.2.3 Effect on State

None.

5.5.2.4 Errors

Table 14 — Error Codes for *GetAppInfoByIDs()*

errorCode	errorDescription	Description
400-499	TBD	See clause 3 in the UPnP Device Architecture [1].
500-599	TBD	See clause 3 in the UPnP Device Architecture [1].
600-699	TBD	See clause 3 in the UPnP Device Architecture [1].
701	Invalid ID	One or more of the *@id*s in the *AppIDs* are not valid.
702	Too many IDs	Too many *@id*s specified in the *AppIDs*.

5.5.3 *GetSupportedTargetFields()*

This required action enables a Screen Control Point to retrieve the *SupportedTargetFields* state variable. This action is used to list the values that are allowed to be used for the

TargetFields input argument in the *GetAppIDList()* action (see subclause 5.5.4) on the Screen Device.

5.5.3.1 Arguments

Table 15 — Arguments for *GetSupportedTargetFields()*

Argument	Direction	Related State Variable
SupportedTargetFields	*OUT*	*SupportedTargetFields*

5.5.3.2 Dependency on State

None.

5.5.3.3 Effect on State

None.

5.5.3.4 Errors

Table 16 — Error Codes for *GetSupportedTargetFields()*

errorCode	errorDescription	Description
400-499	TBD	See clause 3 in the UPnP Device Architecture [1].
500-599	TBD	See clause 3 in the UPnP Device Architecture [1].
600-699	TBD	See clause 3 in the UPnP Device Architecture [1].

5.5.4 *GetAppIDList()*

This required action enables a Screen Control Point to retrieve a CSV list of the *@id* values of specific <*application*> elements in the *AppInfoList* state variable on the Screen Device. The <*application*> elements of the returned *application@id* values shall have a sub-string(s) matched to the specified string by the *Target* input argument (see subclause 5.3.6) among any of their sub-elements specified by the *TargetFields* input argument.

The allowed values for the *TargetFields* input argument are listed in the *SupportedTargetFields* state variable (see subclauses 5.3.5 and 5.3.7).

5.5.4.1 Arguments

Table 17 — Arguments for *GetAppIDList()*

Argument	Direction	Related State Variable
Target	*IN*	*A_ARG_TYPE_Target*
TargetFields	*IN*	*A_ARG_TYPE_TargetFields*
AppIDs	*OUT*	*A_ARG_TYPE_AppIDs*

5.5.4.2 Dependency on State

None.

5.5.4.3 Effect on State

None.

5.5.4.4 Errors

Table 18 — Error Codes for *GetAppIDList()*

errorCode	errorDescription	Description
400-499	TBD	See clause 3 in the UPnP Device Architecture [1].
500-599	TBD	See clause 3 in the UPnP Device Architecture [1].
600-699	TBD	See clause 3 in the UPnP Device Architecture [1].

errorCode	errorDescription	Description
703	Invalid TargetFields	*TargetFields* contains unsupported values.

5.5.5 *GetRunningAppList()*

This conditionally required action shall be supported if the *START feature* is supported (see subclauses 5.2.1 and 5.3.1), and allowed otherwise. This action enables a Screen Control Point to retrieve a list of running applications, i.e., the *RunningAppList* state variable, on the Screen Device (see subclause 5.3.8).

5.5.5.1 Arguments

Table 19 — Arguments for *GetRunningAppList()*

Argument	Direction	Related State Variable
RunningAppList	*OUT*	*RunningAppList*

5.5.5.2 Dependency on State

None.

5.5.5.3 Effect on State

None.

5.5.5.4 Errors

Table 20 — Error Codes for *GetRunningAppList()*

errorCode	errorDescription	Description
400-499	TBD	See clause 3 in the UPnP Device Architecture [1].
500-599	TBD	See clause 3 in the UPnP Device Architecture [1].
600-699	TBD	See clause 3 in the UPnP Device Architecture [1].

5.5.6 *GetRunningStatus()*

This conditionally required action shall be supported if the *START feature* is supported (see subclauses 5.2.1 and 5.3.1), and allowed otherwise. This action enables a Screen Control Point to retrieve the running status of applications specified by the *AppIDs* argument.

5.5.6.1 Arguments

- *AppIDs*: Specifies applications to retrieve their running status. See subclause 5.3.2.
- *RunningStatus*: an XML fragment of the *AppInfoList* state variable (see subclauses 5.3.3 and 5.3.4). It shall contain the <*application*> elements (of which their *@id* values are identical to the values of the *AppIDs* input argument), their attributes, and their <*runningStatus*> sub-elements. If any value of the *AppIDs* input argument is not valid, it shall return either error code 701 or respond with a *RunningStatus* output argument containing <*application*> elements corresponding only to the valid values of the *AppIDs* input argument. The number of <*application*> elements of the *RunningStatus* output argument shall be less than or equal to the number of *application@id*s included in the *AppIDs* input argument.

Table 21 — Arguments for *GetRunningStatus()*

Argument	Direction	Related State Variable
AppIDs	*IN*	*A_ARG_TYPE_AppIDs*
RunningStatus	*OUT*	*A_ARG_TYPE_AppInfo*

5.5.6.2 Dependency on State

None.

5.5.6.3 Effect on State

None.

5.5.6.4 Errors

Table 22 — Error Codes for *GetRunningStatus()*

errorCode	errorDescription	Description
400-499	TBD	See clause 3 in the UPnP Device Architecture [1].
500-599	TBD	See clause 3 in the UPnP Device Architecture [1].
600-699	TBD	See clause 3 in the UPnP Device Architecture [1].
701	Invalid ID	One or more of the *@id*s in the *AppIDs* are not valid.
702	Too many IDs	Too many *@id*s specified in the *AppIDs*.

5.5.7 *StartAppByID()*

This conditionally required action shall be supported if the *START feature* is supported (see subclauses 5.2.1 and 5.3.1), and allowed otherwise. This action runs an application of which its information is contained in the *AppInfoList* state variable on the Screen Device when successfully accepted. In addition, this action can be used to provide the *StartParameters* on an application in a status of "*Transitioning_Pending_Input*" or "*Running*".

5.5.7.1 Arguments

- *AppID*: Specifies the application to be started. See subclause 5.3.2. This argument shall contain only a single *application@id* value.
- *StartParameters*: see subclause 5.3.11.

Table 23 — Arguments for *StartAppByID()*

Argument	Direction	Related State Variable
AppID	*IN*	*A_ARG_TYPE_AppIDs*
StartParameters	*IN*	*A_ARG_TYPE_Parameters*

5.5.7.2 Dependency on State

None.

5.5.7.3 Effect on State

This action will affect the *AppInfoList* state variable. This action changes the *AppInfoList::application::runningStatus* value of "*Inactive*" to "*Running*". If it takes a noticeable amount of time before a human user is actually served by an application, it may temporarily enter "*Transitioning*" status before entering "*Running*".

If a Screen Device requests user input during starting an application, it enters the "*Transitioning_Pending_Input*" before entering "*Running*". Once the user input is provided (possibly by invoking this action again with a *StartParameters*), the status will change to "*Running*".

Consequently, this action will also affect the *RunningAppList* state variable. The *AppInfoList::application@id* of the application will be included in the *RunningAppList* state variable when its *AppInfoList::application::runningStatus* is set to "*Running*".

5.5.7.4 Errors

Table 24 — Error Codes for *StartAppByID()*

Error Code	Error Description	Description
400-499	TBD	See clause 3 in the UPnP Device Architecture [1].
500-599	TBD	See clause 3 in the UPnP Device Architecture [1].
600-699	TBD	See clause 3 in the UPnP Device Architecture [1].
701	Invalid ID	One or more of the *@id*s in the *AppIDs* are not valid.
702	Too many IDs	Too many *@id*s specified in the *AppIDs*, i.e. more than one.
704	Invalid Parameter	Application cannot start due that the specified parameter is invalid.
705	Application is running	Application's <*runningStatus*> is already "*Running*" and no *StartParameters* specified.
706	Rejected	This request is rejected, e.g., by Screen Device implementation or end user.
712	No such Application is installed	Any of applications' <*installationStatus*>s is not "*Installed*", and installation is required to start.

5.5.8 *StartAppbyURI()*

This conditionally required action shall be supported if the *START feature* is supported (see subclauses 5.2.1 and 5.3.1), and allowed otherwise. This action runs an application by using a URI on the Screen Device when successfully accepted. In addition, this action can be used to provide the *StartParameters* on an application in a status of "*Transitioning_Pending_Input*".

5.5.8.1 Arguments

- *StartURI*: Provides the <*startURI*> to start an application. See subclause 5.3.10.
- *AppInfo*: an XML fragment of the *AppInfoList* state variable (see subclauses 5.3.3 and 5.3.4). Provides the additional information for the application to be started. The <*friendlyName*> is required.
- *StartParameters*: see subclause 5.3.11.
- *AppID*: Provides the newly-assigned *application@id* value, or the *@id* value of the <*application*> of which its <*startURI*> is identical to the *StartURI* input argument. This argument shall contain only a single *application@id* value.

When an application can be started without being installed, by using a URI, i.e. a *Web-application*, and the Screen Device does not have the <*startURI*> in its *AppInfoList* state variable, then this action shall be invoked to start the application on the Screen Device. If the action is successfully accepted, the Screen Device shall follow the procedures as below:

- create a new <*application*> element in its *AppInfoList* state variable. The new <*application*> shall include the <*startURI*> and the additional information provided by the *StartURI* and *AppInfo* input arguments.
- assign a new value for the *application@id* attribute.
- return the *AppID* output argument of the newly-assigned *application@id* value.

Table 25 — Arguments for *StartAppByURI()*

Argument	Direction	Related State Variable
StartURI	*IN*	*A_ARG_TYPE_URI*
AppInfo	*IN*	*A_ARG_TYPE_AppInfo*
StartParameters	*IN*	*A_ARG_TYPE_Parameters*
AppID	*OUT*	*A_ARG_TYPE_AppIDs*

5.5.8.2 Dependency on State

None.

5.5.8.3 Effect on State

This action will affect the *AppInfoList* state variable. This action adds a new <*application*> element in the *AppInfoList* state variable if there is no <*application*> element of which its <*startURI*> is identical to the *StartURI* input argument. Its *AppInfoList::application::runningStatus* value will be "*Running*". If it takes a noticeable amount of time before a human user is actually served by an application, it may temporarily enter "*Transitioning*" status before entering "*Running*".

If a Screen Device requests user input during starting an application, it enters the "*Transitioning_Pending_Input*" before entering "*Running*". Once the user input is provided (possibly by invoking this action again with a *StartParameters*), the status will change to "*Running*".

Consequently, this action will also affect the *RunningAppList* state variable. The *AppInfoList::application@id* of the application will be included in the *RunningAppList* state variable when its *AppInfoList::application::runningStatus* is set to "*Running*".

5.5.8.4 Errors

Table 26 — Error Codes for *StartAppByURI()*

Error Code	Error Description	Description
400-499	TBD	See clause 3 in the UPnP Device Architecture [1].
500-599	TBD	See clause 3 in the UPnP Device Architecture [1].
600-699	TBD	See clause 3 in the UPnP Device Architecture [1].
704	Invalid Parameter	Application cannot start due that the specified parameter is invalid
705	Application is running	Application is already running and no *StartParameters* specified.
706	Rejected	This request is rejected, e.g., by Screen Device implementation or end user.
707	Invalid URI	The *StartURL* is invalid, e.g., does not represent an endpoint that can be started.
708	Invalid AppInfo	AppInfo is invalid.
717	Installation required	The application is required to be installed before it can be started.

5.5.9 *StopApp()*

This conditionally required action shall be supported if the *START feature* is supported (see subclauses 5.2.1 and 5.3.1), and allowed otherwise. This action stops applications specified by the *AppIDs* input argument on the Screen Device when successfully accepted.

5.5.9.1 Arguments

- *AppIDs*: Specifies the applications to be stopped. See subclause 5.3.2.
- *StoppedAppIDs:* The list of the *application@id* values of the stopped applications among the requested ones shall be returned with the *StoppedAppIDs* output argument. If any value of the *AppIDs* input argument is not valid, it shall return either error code 701 or respond with a *StoppedAppIDs* output argument containing *application@id*s which are valid and stopped by this action invocation. The output argument shall have an empty string when all values of the *AppIDs* input argument are valid but no application is stopped by this action invocation.

Table 27 — Arguments for *StopApp()*

Argument	Direction	Related State Variable
AppIDs	*IN*	*A_ARG_TYPE_AppIDs*
StoppedAppIDs	*OUT*	*A_ARG_TYPE_AppIDs*

5.5.9.2 Dependency on State

None.

5.5.9.3 Effect on State

This action will affect the *AppInfoList* state variable. This action changes the *AppInfoList::application::runningStatus* value to "*Inactive*".

Consequently, this action will also affect the *RunningAppList* state variable. The *AppInfoList::application@id* of the application will be excluded from the *RunningAppList* state variable when its *AppInfoList::application::runningStatus* is set to any other value than "*Running*".

5.5.9.4 Errors

Table 28 — Error Codes for *StopApp()*

Error Code	Error Description	Description
400-499	TBD	See clause 3 in the UPnP Device Architecture [1].
500-599	TBD	See clause 3 in the UPnP Device Architecture [1].
600-699	TBD	See clause 3 in the UPnP Device Architecture [1].
701	Invalid ID	One or more of the *@id*s in the *AppIDs* are not valid.
706	Rejected	This request is rejected, e.g., by Screen Device implementation or end user.
709	No such Application is running	Any of applications' <*runningStatus*>s is not "*Running*".
710	Not stoppable application	Any of applications listed by the *@id*s in the *AppIDs* cannot be stopped.

5.5.10 *InstallAppByID()*

This conditionally required action shall be supported if the *INSTALL feature* is supported (see subclauses 5.2.1 and 5.3.1), and allowed otherwise. This action installs an application using information contained in the *AppInfoList* state variable on the Screen Device when successfully accepted. In addition, this action can be used to provide the *InstallParameters* for an application in a status of "*Installing_Pending_Input*". Moreover, this action can also be used to update an installed application to its latest version if exists.

5.5.10.1 Arguments

- *AppID*: Specifies the application to be installed. See subclause 5.3.2. This argument shall contain only a single *application@id* value.
- *InstallParameters*: See subclause 5.3.11.

Table 29 — Arguments for *InstallAppByID()*

Argument	Direction	Related State Variable
AppID	*IN*	*A_ARG_TYPE_AppIDs*
InstallParameters	*IN*	*A_ARG_TYPE_Parameters*

5.5.10.2 Dependency on State

None.

5.5.10.3 Effect on State

This action will affect the *AppInfoList* state variable. This action changes the *AppInfoList::application::installationStatus* value of the "*Not_Downloaded*" or "*Not_Installed*" status to the "*Installed*" status. If it takes a noticeable amount of time before an application is completely downloaded, it may temporarily enter "*Downloading*" status before entering "*Installed*". If it takes a noticeable amount of time before an application is completely installed

after the completion of downloading, it may temporarily enter "*Installing*" status before entering to "*Installed*".

If a Screen Device requests user input during downloading or installing an application, it enters the "*Downloading_Pending_Input*" or "*Installing_Pending_Input*" status before entering the "*Not_Installed*" or "*Installed*" status respectively. Once the user input is provided (possibly by invoking this action again with an *InstallParameters* input argument), the status will change to the "*Not_Installed*" (and to next status sequentially) or "*Installed*" status respectively.

5.5.10.4 Errors

Table 30 — Error Codes for *InstallAppByID()*

errorCode	errorDescription	Description
400-499	TBD	See clause 3 in the UPnP Device Architecture [1].
500-599	TBD	See clause 3 in the UPnP Device Architecture [1].
600-699	TBD	See clause 3 in the UPnP Device Architecture [1].
701	Invalid ID	One or more of the *@id*s in the *AppIDs* are not valid.
702	Too many IDs	Too many *@id*s specified in the *AppIDs*, i.e. more than one.
704	Invalid Parameter	Application cannot be installed due that the specified parameter is invalid
711	Application is installed	Application's <*installationStatus*> is already "*Installed*" and it is the latest version.
706	Rejected	This request is rejected, e.g., by Screen Device implementation or end user.
715	Not enough storage	Storage is not enough to install the application.
716	Exceeding download limit	The application size exceeds the download limit.

5.5.11 *InstallAppByURI()*

This conditionally required action shall be supported if the *INSTALL feature* is supported (see subclauses 5.2.1 and 5.3.1), and allowed otherwise. This action installs an application by using a URI on the Screen Device when successfully accepted. In addition, this action can be used to provide the *InstallParameters* on an application in a status of "*Installing_Pending_Input*".

5.5.11.1 Arguments

- *InstallationURI*: Provides <*installationURI*> to install an application. See subclause 5.3.10.
- *AppInfo*: an XML fragment of the *AppInfoList* state variable (see subclauses 5.3.3 and 5.3.4). Provides the additional information for the application to be installed. The <*friendlyName*> is required.
- *InstallParameters*: See subclause 5.3.11.
- *AppID*: Provides the newly-assigned *application@id* value or the *@id* value of the <*application*> of which its <*installationURI*> is identical to the *InstallationURI* input argument. This argument shall contain only a single *application@id* value.

When the Screen Device does not have the <*installationURI*> in its *AppInfoList* state variable, then this action shall be invoked to install the application on the Screen Device. If the action is successfully accepted, the Screen Device shall follow the procedures as below:

- create a new <*application*> element in its *AppInfoList* state variable. The new <*application*> shall include the <*installationURI*> and the additional information provided by the *InstallationURI* and *AppInfo* input arguments.
- assign a new value for the *application@id* attribute.
- return the *AppID* output argument of the newly-assigned *application@id* value.

Table 31 — Arguments for *InstallAppByURI()*

Argument	Direction	Related State Variable
InstallationURI	*IN*	*A_ARG_TYPE_URI*
AppInfo	*IN*	*A_ARG_TYPE_AppInfo*
InstallParameters	*IN*	*A_ARG_TYPE_Parameters*
AppID	*OUT*	*A_ARG_TYPE_AppIDs*

5.5.11.2 Dependency on State

None.

5.5.11.3 Effect on State

This action will affect the *AppInfoList* state variable. This action adds a new <*application*> element in the *AppInfoList* state variable if there is no <*application*> element of which its <*installationURI*> is identical to the *InstallationURI* input argument. Its *AppInfoList::application::installationStatus* value will be "*Installed*". If it takes a noticeable amount of time before an application is completely downloaded, it may temporarily enter "*Downloading*" status before entering "*Installed*". If it takes a noticeable amount of time before an application is completely installed after the completion of downloading, it may temporarily enter "*Installing*" status before entering "*Installed*".

If a Screen Device requests user input during downloading or installing an application, it enters the "*Downloading_Pending_Input*" or "*Installing_Pending_Input*" status before entering the "*Not_Installed*" or "*Installed*" status respectively. Once the user input is provided (possibly by invoking this action again with an *InstallParameters* input argument), the status will change to the "*Not_Installed*" (and to next status sequentially) or "*Installed*" status respectively.

5.5.11.4 Errors

Table 32 — Error Codes for *InstallAppByURI()*

errorCode	errorDescription	Description
400-499	TBD	See clause 3 in the UPnP Device Architecture [1].
500-599	TBD	See clause 3 in the UPnP Device Architecture [1].
600-699	TBD	See clause 3 in the UPnP Device Architecture [1].
704	Invalid Parameter	Application can not start due that the specified parameter is invalid
711	Application is installed	Application is already installed..
706	Rejected	This request is rejected, e.g., by Screen Device implementation or end user.
707	Invalid URI	The *InstallURL* is invalid, e.g. does not represent an endpoint that can be installed.
708	Invalid AppInfo	AppInfo is invalid.
715	Not enough storage	Storage is not enough to install the application.
716	Exceeding download limit	The application size exceeds the download limit.

5.5.12 *UninstallApp()*

This conditionally required action shall be supported if the *INSTALL feature* is supported (see subclauses 5.2.1 and 5.3.1), and allowed otherwise. This action uninstalls applications specified by the *AppIDs* input argument on the Screen Device when successfully accepted.

5.5.12.1 Arguments

- *AppIDs*: Specifies the applications to be uninstalled. See subclause 5.3.2.

- *UninstalledAppIDs:* The list of the *application@id* values of the uninstalled applications among the requested ones shall be returned with the *UninstalledAppIDs* output argument. If any value of the *AppIDs* input argument is not valid, it shall return either error code 701 or respond with a *UninstalledAppIDs* output argument containing *application@id*s which are valid and uninstalled by this action invocation. The output argument shall have an empty string when all values of the *AppIDs* input argument are valid but no application is uninstalled by this action invocation.

Table 33 — Arguments for *UninstallApp()*

Argument	Direction	Related State Variable
AppIDs	*IN*	*A_ARG_TYPE_AppIDs*
UninstalledAppIDs	*OUT*	*A_ARG_TYPE_AppIDs*

5.5.12.2 Dependency on State

None.

5.5.12.3 Effect on State

This action will affect the *AppInfoList* state variable. This action changes the *AppInfoList::application::installationStatus* value to "*Not_Installed*".

5.5.12.4 Errors

Table 34 — Error Codes for *UninstallApp()*

errorCode	errorDescription	Description
400-499	TBD	See clause 3 in the UPnP Device Architecture [1].
500-599	TBD	See clause 3 in the UPnP Device Architecture [1].
600-699	TBD	See clause 3 in the UPnP Device Architecture [1].
701	Invalid ID	One or more of the *@id*s in the *AppIDs* are not valid.
706	Rejected	This request is rejected, e.g., by Screen Device implementation or end user.
712	No such Application is installed	Any of applications' <*installationStatus*>s is not "*Installed*".
713	Not uninstallable application	Any of applications listed by the *@id*s in the *AppIDs* cannot be uninstalled.

5.5.13 *GetInstallationStatus()*

This conditionally required action shall be supported if the *INSTALL feature* is supported (see subclauses 5.2.1 and 5.3.1), and allowed otherwise. This action enables a Screen Control Point to retrieve the installation status of applications specified by the *AppIDs* input argument.

5.5.13.1 Arguments

- *AppIDs*: Specifies the applications to retrieve its installation status. See subclause 5.3.2.
- *InstallationStatus*: an XML fragment of the *AppInfoList* state variable (see subclauses 5.3.3 and 5.3.4). It shall contain <*application*> elements (of which their *@id* values are identical to the values of the *AppIDs* input argument), their attributes, and their <*installationStatus*>, <*downloadingProgress*> and <*installationProgress*> sub-elements. If any value of the *AppIDs* input argument is not valid, it shall return either error code 701 or respond with a *InstallationStatus* output argument containing <*application*> elements corresponding only to the valid values of the *AppIDs* input argument. The number of <*application*> elements of the *InstallationStatus* output argument shall be less than or equal to the number of *application@id*s included in the *AppIDs* input argument.

Table 35 — Arguments for *GetInstallationStatus()*

Argument	Direction	Related State Variable
AppIDs	*IN*	*A_ARG_TYPE_AppIDs*
InstallationStatus	*OUT*	*A_ARG_TYPE_AppInfo*

5.5.13.2 Dependency on State

None.

5.5.13.3 Effect on State

None.

5.5.13.4 Errors

Table 36 — Error Codes for *GetInstallationStatus()*

errorCode	errorDescription	Description
400-499	TBD	See clause 3 in the UPnP Device Architecture [1].
500-599	TBD	See clause 3 in the UPnP Device Architecture [1].
600-699	TBD	See clause 3 in the UPnP Device Architecture [1].
701	Invalid ID	One or more of the *@id*s in the *AppIDs* are not valid.
702	Too many IDs	Too many *@id*s specified in the *AppIDs*.

5.5.14 *GetAppConnectionInfo()*

This conditionally required action shall be supported if the *CONNECT feature* is supported (see subclauses 5.2.1 and 5.3.1), and allowed otherwise. This action enables a Screen Control Point to retrieve the app-to-app connection information of applications specified by the *AppIDs* input argument.

5.5.14.1 Arguments

- *AppIDs*: Specifies the application to retrieve their app-to-app connection information. See subclause 5.3.2.
- *ConnectionInfo*: an XML fragment of the *AppInfoList* state variable (see subclauses 5.3.3 and 5.3.4). It shall contain <*application*> elements (of which their *@id* values are identical to the values of the *AppIDs* input arguments), their attributes, and their <*apptoAppInfo*> sub-elements if supported. If the <*apptoAppInfo*> sub-element of an <*application*> element is not supported, then it shall contain <*application*> and its attribute only. If any value of the *AppIDs* input argument is not valid, it shall return either error code 701 or respond with a *ConnectionInfo* output argument containing <*application*> elements corresponding only to the valid values of the *AppIDs* input argument. The number of <*application*> elements of the *ConnectionInfo* output argument shall be less than or equal to the number of *application@id*s included in the *AppIDs* input argument.

Table 37 — Arguments for *GetAppConnectionInfo()*

Argument	Direction	Related State Variable
AppIDs	*IN*	*A_ARG_TYPE_AppIDs*
ConnectionInfo	*OUT*	*A_ARG_TYPE_AppInfo*

5.5.14.2 Dependency on State

None.

5.5.14.3 Effect on State

None.

5.5.14.4 Errors

Table 38 — Error Codes for *GetAppConnectionInfo()*

errorCode	errorDescription	Description
400-499	TBD	See clause 3 in the UPnP Device Architecture [1].
500-599	TBD	See clause 3 in the UPnP Device Architecture [1].
600-699	TBD	See clause 3 in the UPnP Device Architecture [1].
701	Invalid ID	One or more of the *@id*s in the *AppIDs* are not valid.
709	No such Application is running	Any of applications' <*runningStatus*>s is not "*Running*".

5.5.15 *ConnectApptoApp()*

This conditionally required action shall be supported if the *CONNECT feature* is supported (see subclauses 5.2.1 and 5.3.1), and allowed otherwise. This action is used to set up an app-to-app connection with the Screen Device and enable the connection manageable.

5.5.15.1 Arguments

- *AppID*: Specifies the application to set up an app-to-app connection with the Screen Device. See subclause 5.3.2. This argument shall contain only a single *application@id* value.
- *ConnectionID*: The Screen Device shall assign a unique identifier of the app-to-app connection to be set up and return it if the action is successfully accepted. See subclause 5.3.12.

The number of components in the both output arguments shall be identical, and their order shall be correctly matched.

Table 39 — Arguments for *ConnectApptoApp()*

Argument	Direction	Related State Variable
AppID	*IN*	*A_ARG_TYPE_AppIDs*
ConnectionID	*OUT*	*A_ARG_TYPE_ConnectionIDs*

5.5.15.2 Dependency on State

None.

5.5.15.3 Effect on State

None.

5.5.15.4 Errors

Table 40 — Error Codes for *ConnectApptoApp()*

errorCode	errorDescription	Description
400-499	TBD	See clause 3 in the UPnP Device Architecture [1].
500-599	TBD	See clause 3 in the UPnP Device Architecture [1].
600-699	TBD	See clause 3 in the UPnP Device Architecture [1].
701	Invalid ID	One or more of the *@id*s in the *AppIDs* are not valid.
702	Too many IDs	Too many *@id*s specified in the *AppIDs*, i.e. more than one.

5.5.16 *DisconnectApptoApp()*

This conditionally required action shall be supported if the *CONNECT feature* is supported (see subclauses 5.2.1 and 5.3.1), and allowed otherwise. This action is used to tear down an app-to-app connection specified by the *ConnectionIDs* input argument from the Screen Device.

5.5.16.1 Arguments

- *ConnectionIDs*: Specifies the connections to be disconnected. See subclause 5.3.12.
- *DisconnectedConnectionIDs:* The list of the ID values of the disconnected app-to-app connections among the requested ones shall be returned with the *DisconnectedConnectionIDs* output argument. If any value of the *ConnectionIDs* input argument is not valid, it shall return either error code 714 or respond with a *DisconnectedConnectionIDs* output argument containing connections' IDs which are valid and disconnected by this action invocation. The output argument shall have an empty string when all values of the *ConnectionIDs* input argument are valid but no connection is disconnected by this action invocation.

Table 41 — Arguments for *DisconnectApptoApp()*

Argument	Direction	Related State Variable
ConnectionIDs	*IN*	*A_ARG_TYPE_ConnectionIDs*
DisconnectedConnectionIDs	*OUT*	*A_ARG_TYPE_ConnectionIDs*

5.5.16.2 Dependency on State

None.

5.5.16.3 Effect on State

None.

5.5.16.4 Errors

Table 42 — Error Codes for *DisconnectApptoApp()*

errorCode	errorDescription	Description
400-499	TBD	See clause 3 in the UPnP Device Architecture [1].
500-599	TBD	See clause 3 in the UPnP Device Architecture [1].
600-699	TBD	See clause 3 in the UPnP Device Architecture [1].
714	Invalid ConnectionID	One or more the connection IDs in the *ConnectionIDs* are not valid.
706	Rejected	This request is rejected, e.g., by Screen Device implementation or end user.

5.5.17 *GetCurrentConnectionInfo()*

This conditionally required action shall be supported if the *CONNECT feature* is supported (see subclauses 5.2.1 and 5.3.1), and allowed otherwise. This action enables a Screen Control Point to collect the information of all the app-to-app connections which were established by the *ConnectApptoApp()* invocations and the Screen Device is currently supporting.

5.5.17.1 Arguments

- *ConnectionIDs*: Provides the current app-to-app connections' identifiers. See subclause 5.3.12.
- *ConnectionAppID*: Provides the applications' *@id* values corresponding to the app-to-app connections. See subclause 5.3.2.

The number of components in the both output arguments shall be identical, and their order shall be correctly matched.

Table 43 — Arguments for *GetCurrentConnectionInfo()*

Argument	Direction	Related State Variable
ConnectionIDs	*OUT*	*A_ARG_TYPE_ConnectionIDs*

