
ApplicationManagement:2 Service

For UPnP Version 1.0

Status: Standardized DCP (SDCP)

Date: September 30, 2014

Service Template Version 3.0

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

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

© 2014, UPnP Forum. All rights Reserved.

Authors	Company
Clarke Stevens	CableLabs
Wouter van der Beek	Cisco Systems Inc.
Seung R. Yang (Chair)	LG Electronics
Anders Klemets	Microsoft
Nicholas Frame	TP Vision

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

© 2014, UPnP Forum. All rights Reserved.

CONTENTS

1	Scope.....	4
2	Normative references.....	4
3	Terms, definitions, symbols and abbreviations.....	5
4	Notations and Conventions.....	5
5	Service Modelling Definitions.....	5
5.1	Service Type.....	5
5.2	Key Concepts.....	5
5.2.1	AMS features.....	5
5.2.2	SECURITY feature.....	6
5.3	State Variables.....	8
5.3.1	<u>FeatureList</u>	9
5.3.2	<u>A_ARG_TYPE_AppIDs</u>	9
5.3.3	<u>AppInfoList</u>	10
5.3.4	<u>A_ARG_TYPE_AppInfo</u>	13
5.3.5	<u>SupportedTargetFields</u>	14
5.3.6	<u>A_ARG_TYPE_Target</u>	14
5.3.7	<u>A_ARG_TYPE_TargetFields</u>	14
5.3.8	<u>RunningAppList</u>	14
5.3.9	<u>TransitioningApps</u>	14
5.3.10	<u>A_ARG_TYPE_URI</u>	14
5.3.11	<u>A_ARG_TYPE_Parameters</u>	14
5.3.12	<u>A_ARG_TYPE_ConnectionIDs</u>	15
5.4	Eventing and Moderation.....	15
5.5	Actions.....	15
5.5.1	<u>GetFeatureList()</u>	16
5.5.2	<u>GetAppInfoByIDs()</u>	17
5.5.3	<u>GetSupportedTargetFields()</u>	17
5.5.4	<u>GetAppIDList()</u>	18
5.5.5	<u>GetRunningAppList()</u>	19
5.5.6	<u>GetRunningStatus()</u>	19
5.5.7	<u>StartAppByID()</u>	20
5.5.8	<u>StartAppbyURI()</u>	21
5.5.9	<u>StopApp()</u>	22
5.5.10	<u>InstallAppByID()</u>	23
5.5.11	<u>InstallAppByURI()</u>	24
5.5.12	<u>UninstallApp()</u>	25
5.5.13	<u>GetInstallationStatus()</u>	26
5.5.14	<u>GetAppConnectionInfo()</u>	27
5.5.15	<u>ConnectApptoApp()</u>	28
5.5.16	<u>DisconnectApptoApp()</u>	28
5.5.17	<u>GetCurrentConnectionInfo()</u>	29
5.5.18	Non-Standard Actions Implemented by a UPnP Vendor.....	30
5.5.19	Common Error Codes.....	30
6	XML Service Description.....	31
7	Test.....	37

Table 1 — <i>AMS features</i>	6
Table 2 — Error Codes for <i>Action Level Access</i>	7
Table 3 — <i>ApplicationManagement Roles</i>	7
Table 4 — Action to <i>Role</i> Permission Mapping	8
Table 5 — State variables	9
Table 6 — Allowed values for <i>function</i> element.....	12
Table 7 — Allowed values for <i>connectionAddress</i> element.....	12
Table 8 — Required fields for <i>SupportedTargetFields</i>	14
Table 9 — Event moderation	15
Table 10 — Actions	16
Table 11 — Arguments for <i>GetFeatureList()</i>	16
Table 12 — Error Codes for <i>GetFeatureList()</i>	17
Table 13 — Arguments for <i>GetAppInfoByIDs()</i>	17
Table 14 — Error Codes for <i>GetAppInfoByIDs()</i>	17
Table 15 — Arguments for <i>GetSupportedTargetFields()</i>	18
Table 16 — Error Codes for <i>GetSupportedTargetFields()</i>	18
Table 17 — Arguments for <i>GetAppIDList()</i>	18
Table 18 — Error Codes for <i>GetAppIDList()</i>	18
Table 19 — Arguments for <i>GetRunningAppList()</i>	19
Table 20 — Error Codes for <i>GetRunningAppList()</i>	19
Table 21 — Arguments for <i>GetRunningStatus()</i>	19
Table 22 — Error Codes for <i>GetRunningStatus()</i>	20
Table 23 — Arguments for <i>StartAppByID()</i>	20
Table 24 — Error Codes for <i>StartAppByID()</i>	21
Table 25 — Arguments for <i>StartAppByURI()</i>	21
Table 26 — Error Codes for <i>StartAppByURI()</i>	22
Table 27 — Arguments for <i>StopApp()</i>	22
Table 28 — Error Codes for <i>StopApp()</i>	23
Table 29 — Arguments for <i>InstallAppByID()</i>	23
Table 30 — Error Codes for <i>InstallAppByID()</i>	24
Table 31 — Arguments for <i>InstallAppByURI()</i>	25
Table 32 — Error Codes for <i>InstallAppByURI()</i>	25
Table 33 — Arguments for <i>UninstallApp()</i>	26
Table 34 — Error Codes for <i>UninstallApp()</i>	26
Table 35 — Arguments for <i>GetInstallationStatus()</i>	27
Table 36 — Error Codes for <i>GetInstallationStatus()</i>	27
Table 37 — Arguments for <i>GetAppConnectionInfo()</i>	27
Table 38 — Error Codes for <i>GetAppConnectionInfo()</i>	28
Table 39 — Arguments for <i>ConnectApptoApp()</i>	28
Table 40 — Error Codes for <i>ConnectApptoApp()</i>	28
Table 41 — Arguments for <i>DisconnectApptoApp()</i>	29
Table 42 — Error Codes for <i>DisconnectApptoApp()</i>	29
Table 43 — Arguments for <i>GetCurrentConnectionInfo()</i>	29

Table 44 — Error Codes for <i>GetCurrentConnectionInfo()</i>	30
Table 45 — Common Error Codes	30

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	<i>GetAppInfoByID()</i> <i>GetSupportedTargetFields()</i> <i>GetAppIDLlist()</i> and the related state variables
<i>START</i>	1	Application-starting related functionalities	<i>StartAppByID()</i> <i>StartAppByURL()</i> <i>StopApp()</i> <i>GetRunningAppList()</i> <i>GetRunningStatus()</i> and the related state variables
<i>INSTALL</i>	1	Application-installing related functionalities	<i>InstallAppByID()</i> <i>InstallAppByURI()</i> <i>UninstallApp()</i> <i>GetInstallationStatus()</i> and the related state variables
<i>CONNECT</i>	1	Application-connecting related functionalities	<i>GetAppConnectionInfo()</i> <i>ConnectAppToApp()</i> <i>DisconnectAppToApp()</i> <i>GetCurrentConnectionInfo()</i> and the related state variables
<i>SECURITY</i>	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
<i>Public</i>	<i>CR</i>
<i>Basic</i>	<i>CR</i>
<i>AM_SuperUser</i>	<i>CR</i>
<i>Admin</i>	<i>CR</i>
Conditionally required if the <i>SECURITY feature</i> 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
<u>GetFeatureList()</u>	Non-Restrictable	<u>YES</u>	<u>YES</u>	<u>YES</u>
<u>GetAppInfoByIDs()</u>	Restrictable	<u>NO</u>	<u>YES</u>	<u>YES</u>
<u>GetSupportedTargetFields()</u>	Non-Restrictable	<u>YES</u>	<u>YES</u>	<u>YES</u>
<u>GetAppIDList()</u>	Restrictable	<u>NO</u>	<u>YES</u>	<u>YES</u>
<u>GetRunningAppList()</u>	Restrictable	<u>NO</u>	<u>YES</u>	<u>YES</u>
<u>GetRunningStatus()</u>	Restrictable	<u>NO</u>	<u>YES</u>	<u>YES</u>
<u>StartAppByID()</u>	Restrictable	<u>NO</u>	<u>YES</u>	<u>YES</u>
<u>StartAppByURI()</u>	Restrictable	<u>NO</u>	<u>YES</u>	<u>YES</u>
<u>StopApp()</u>	Restrictable	<u>NO</u>	<u>YES</u>	<u>YES</u>
<u>InstallAppByID()</u>	Restrictable	<u>NO</u>	<u>NO</u>	<u>YES</u>
<u>InstallAppByURL()</u>	Restrictable	<u>NO</u>	<u>NO</u>	<u>YES</u>
<u>UninstallApp()</u>	Restrictable	<u>NO</u>	<u>NO</u>	<u>YES</u>
<u>GetInstallationStatus()</u>	Restrictable	<u>NO</u>	<u>YES</u>	<u>YES</u>
<u>GetAppConnectionInfo()</u>	Restrictable	<u>NO</u>	<u>YES</u>	<u>YES</u>
<u>ConnectApptoApp()</u>	Restrictable	<u>NO</u>	<u>YES</u>	<u>YES</u>
<u>DisconnectApptoApp()</u>	Restrictable	<u>NO</u>	<u>YES</u>	<u>YES</u>
<u>GetCurrentConnectionInfo()</u>	Restrictable	<u>NO</u>	<u>YES</u>	<u>YES</u>

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
<i>FeatureList</i>	<i>R</i>	<i>string</i>	CSV(<i>string</i>) See subclause 5.3.1		
<i>A_ARG_TYPE_AppIDs</i>	<i>R</i>	<i>string</i>	CSV(<i>string</i>) See subclause 5.3.2		
<i>AppInfoList</i>	<i>R</i>	<i>string</i>	<i>AppInfoList XML Document</i> See subclause 5.3.3		
<i>A_ARG_TYPE_AppInfo</i>	<i>R</i>	<i>string</i>	<i>XML fragment of AppInfoList XML Document</i> See subclause 5.3.4		
<i>SupportedTargetFields</i>	<i>R</i>	<i>string</i>	CSV(<i>string</i>) See subclause 5.3.5		
<i>A_ARG_TYPE_Target</i>	<i>R</i>	<i>string</i>	See subclause 5.3.6		
<i>A_ARG_TYPE_TargetFields</i>	<i>R</i>	<i>string</i>	CSV(<i>string</i>) See subclause 5.3.7		
<i>RunningAppList</i>	<i>CR</i> ^c	<i>string</i>	CSV(<i>string</i>) See subclause 5.3.8		
<i>TransitioningApps</i>	<i>CR</i> ^c	<i>string</i>	CSV(<i>string</i>) See subclause 5.3.9		
<i>A_ARG_TYPE_URI</i>	<i>CR</i> ^c	<i>string</i>	See subclause 5.3.10		
<i>A_ARG_TYPE_Parameter</i>	<i>CR</i> ^c	<i>string</i>	See subclause 5.3.11		
<i>A_ARG_TYPE_ConnectionIDs</i>	<i>CR</i> ^b	<i>string</i>	CSV(<i>string</i>) See subclause 5.3.12		
<i>Non-standard state variables implemented by a UPnP vendor go here</i>	<i>X</i>	<i>TBD</i>	<i>TBD</i>	<i>TBD</i>	<i>TBD</i>

^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 *vendor-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

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

Table 7 — Allowed values for *connectionAddress* element

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

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 <i>AppInfoList</i>	R/A
<i>friendlyName</i>	<i>application::friendlyName</i>	<i>R</i>
<i>matchingProtocolName</i>	<i>application::apptoAppInfo::matchingProtocolName</i>	<i>R</i>

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
<u>FeatureList</u>	<u>NO</u>	<u>NO</u>			
<u>A_ARG_TYPE_AppIDs</u>	<u>NO</u>	<u>NO</u>			
<u>AppInfoList</u>	<u>NO</u>	<u>NO</u>			
<u>A_ARG_TYPE_AppInfo</u>	<u>NO</u>	<u>NO</u>			
<u>SupportedTargetFields</u>	<u>NO</u>	<u>NO</u>			
<u>A_ARG_TYPE_Target</u>	<u>NO</u>	<u>NO</u>			
<u>A_ARG_TYPE_TargetFields</u>	<u>NO</u>	<u>NO</u>			
<u>RunningAppList</u>	<u>YES</u>	<u>YES</u>	0.2		
<u>TransitioningApps</u>	<u>YES</u>	<u>YES</u>	0.2		
<u>A_ARG_TYPE_URI</u>	<u>NO</u>	<u>NO</u>			
<u>A_ARG_TYPE_Parameters</u>	<u>NO</u>	<u>NO</u>			
<u>A_ARG_TYPE_ConnectionIDs</u>	<u>NO</u>	<u>NO</u>			
<i>Non-standard state variables implemented by a UPnP vendor go here</i>	<i>TBD</i>	<i>TBD</i>	<i>TBD</i>	<i>TBD</i>	<i>TBD</i>

^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
<u>GetFeatureList()</u>	<u>CR</u> ^c	<u>A</u>
<u>GetAppInfoByIDs()</u>	<u>R</u>	<u>R</u>
<u>GetSupportedTargetFields()</u>	<u>R</u>	<u>A</u>
<u>GetAppIDList()</u>	<u>R</u>	<u>R</u>
<u>GetRunningAppList()</u>	<u>CR</u> ^d	<u>A</u>
<u>GetRunningStatus()</u>	<u>CR</u> ^d	<u>A</u>
<u>StartAppByID()</u>	<u>CR</u> ^d	<u>A</u>
<u>StartAppByURI()</u>	<u>CR</u> ^d	<u>A</u>
<u>StopApp()</u>	<u>CR</u> ^d	<u>A</u>
<u>InstallAppByID()</u>	<u>CR</u> ^c	<u>A</u>
<u>InstallAppByURL()</u>	<u>CR</u> ^c	<u>A</u>
<u>UninstallApp()</u>	<u>CR</u> ^c	<u>A</u>
<u>GetInstallationStatus()</u>	<u>CR</u> ^c	<u>A</u>
<u>GetAppConnectionInfo()</u>	<u>CR</u> ^c	<u>A</u>
<u>ConnectApptoApp()</u>	<u>CR</u> ^c	<u>A</u>
<u>DisconnectApptoApp()</u>	<u>CR</u> ^c	<u>A</u>
<u>GetCurrentConnectionInfo()</u>	<u>CR</u> ^c	<u>A</u>
<i>Non-standard actions implemented by an UPnP vendor go here.</i>	<u>X</u>	<u>X</u>
<p>^a For a device this column indicates whether the action shall be implemented or not, where <u>R</u> = required, <u>A</u> = allowed, <u>CR</u> = conditionally required, <u>CA</u> = conditionally allowed, <u>X</u> = Non-standard, add <u>-D</u> when deprecated (e.g., <u>R-D</u>, <u>A-D</u>).</p> <p>^b For a control point this column indicates whether a control point shall be capable of invoking this action, where <u>R</u> = required, <u>A</u> = allowed, <u>CR</u> = conditionally required, <u>CA</u> = conditionally allowed, <u>X</u> = Non-standard, add <u>-D</u> when deprecated (e.g., <u>R-D</u>, <u>A-D</u>).</p> <p>^c <u>CR</u> = conditionally required. See referenced subclause for implementation requirements.</p> <p>^d <u>CR</u> = conditionally required. In fact required since the condition is required for this specification. See referenced subclause for implementation requirements.</p>		

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
<u>FeatureList</u>	<u>OUT</u>	<u>FeatureList</u>

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@ids* included in the *AppIDs* input argument.

Table 13 — Arguments for *GetAppInfoByIds()*

Argument	Direction	Related State Variable
<i>AppIDs</i>	<i>IN</i>	<i>A_ARG_TYPE_AppIDs</i>
<i>AppInfo</i>	<i>OUT</i>	<i>A_ARG_TYPE_AppInfo</i>

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 <i>@ids</i> in the <i>AppIDs</i> are not valid.
702	Too many IDs	Too many <i>@ids</i> specified in the <i>AppIDs</i> .

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
<u>SupportedTargetFields</u>	<u>OUT</u>	<u>SupportedTargetFields</u>

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
<u>Target</u>	<u>IN</u>	<u>A_ARG_TYPE_Target</u>
<u>TargetFields</u>	<u>IN</u>	<u>A_ARG_TYPE_TargetFields</u>
<u>AppIDs</u>	<u>OUT</u>	<u>A_ARG_TYPE_AppIDs</u>

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@ids](#) 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 <i>@ids</i> in the <i>AppIDs</i> are not valid.
702	Too many IDs	Too many <i>@ids</i> specified in the <i>AppIDs</i> .

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
<i>AppID</i>	<i>IN</i>	<i>A_ARG_TYPE AppIDs</i>
<i>StartParameters</i>	<i>IN</i>	<i>A_ARG_TYPE Parameters</i>

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 <i>@ids</i> in the <i>AppIDs</i> are not valid.
702	Too many IDs	Too many <i>@ids</i> specified in the <i>AppIDs</i> , i.e. more than one.
704	Invalid Parameter	Application cannot start due that the specified parameter is invalid
705	Application is running	Application's <i><runningStatus></i> is already " <i>Running</i> " and no <i>StartParameters</i> 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' <i><installationStatus></i> s is not " <i>Installed</i> ", 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
<i>StartURI</i>	<i>IN</i>	<i>A_ARG_TYPE_URI</i>
<i>AppInfo</i>	<i>IN</i>	<i>A_ARG_TYPE_AppInfo</i>
<i>StartParameters</i>	<i>IN</i>	<i>A_ARG_TYPE_Parameters</i>
<i>AppID</i>	<i>OUT</i>	<i>A_ARG_TYPE_AppIDs</i>

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 <i>StartParameters</i> specified.
706	Rejected	This request is rejected, e.g., by Screen Device implementation or end user.
707	Invalid URI	The <i>StartURL</i> 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@ids* 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
<i>AppIDs</i>	<i>IN</i>	<i>A_ARG_TYPE AppIDs</i>
<i>StoppedAppIDs</i>	<i>OUT</i>	<i>A_ARG_TYPE AppIDs</i>

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 @ids 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 @ids 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 <i>@ids</i> in the <i>AppIDs</i> are not valid.
702	Too many IDs	Too many <i>@ids</i> specified in the <i>AppIDs</i> , 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 <i><installationStatus></i> is already “ <i>Installed</i> ” 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
<i>InstallationURI</i>	<i>IN</i>	<i>A_ARG_TYPE_URI</i>
<i>AppInfo</i>	<i>IN</i>	<i>A_ARG_TYPE_AppInfo</i>
<i>InstallParameters</i>	<i>IN</i>	<i>A_ARG_TYPE_Parameters</i>
<i>AppID</i>	<i>OUT</i>	<i>A_ARG_TYPE_AppIDs</i>

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 <i>InstallURL</i> 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@ids 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
<u>AppIDs</u>	<u>IN</u>	<u>A_ARG_TYPE_AppIDs</u>
<u>UninstalledAppIDs</u>	<u>OUT</u>	<u>A_ARG_TYPE_AppIDs</u>

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 <u>@ids</u> in the <u>AppIDs</u> 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' <u><installationStatus></u> s is not " <u>Installed</u> ".
713	Not uninstallable application	Any of applications listed by the <u>@ids</u> in the <u>AppIDs</u> 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@ids 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 @ids in the AppIDs are not valid.
702	Too many IDs	Too many @ids 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@ids](#) 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 <i>@ids</i> in the <i>AppIDs</i> are not valid.
709	No such Application is running	Any of applications' <i><runningStatus></i> s is not " <i>Running</i> ".

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
<i>AppID</i>	<i>IN</i>	<i>A_ARG_TYPE_AppIDs</i>
<i>ConnectionID</i>	<i>OUT</i>	<i>A_ARG_TYPE_ConnectionIDs</i>

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 <i>@ids</i> in the <i>AppIDs</i> are not valid.
702	Too many IDs	Too many <i>@ids</i> specified in the <i>AppIDs</i> , 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
<u>ConnectionIDs</u>	<u>IN</u>	<u>A_ARG_TYPE_ConnectionIDs</u>
<u>DisconnectedConnectionIDs</u>	<u>OUT</u>	<u>A_ARG_TYPE_ConnectionIDs</u>

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 <u>ConnectionIDs</u> 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
<u>ConnectionIDs</u>	<u>OUT</u>	<u>A_ARG_TYPE_ConnectionIDs</u>

Argument	Direction	Related State Variable
<u>ConnectionAppIDs</u>	<u>OUT</u>	<u>A_ARG_TYPE_AppIDs</u>

5.5.17.2 Dependency on State

None.

5.5.17.3 Effect on State

None.

5.5.17.4 Errors

Table 44 — Error Codes for GetCurrentConnectionInfo()

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].

5.5.18 Non-Standard Actions Implemented by a UPnP Vendor

To facilitate certification, non-standard actions implemented by a UPnP vendor shall be included in the device's service template. The UPnP Device Architecture lists naming requirements for non-standard actions (see clause 2 of the UPnP Device Architecture specification [1]).

5.5.19 Common Error Codes

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

Table 45 — Common Error Codes

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 <u>@ids</u> in the <u>AppIDs</u> are not valid.
702	Too many IDs	Too many <u>@ids</u> specified in the <u>AppIDs</u> .
703	Invalid TargetFields	<u>TargetFields</u> contains unsupported values.
704	Invalid Parameter	The specified parameter is invalid
705	Application is running	Application is already running.
706	Rejected	This request is rejected, e.g., by Screen Device implementation or end user.
707	Invalid URI	The specified URL is invalid.
708	Invalid AppInfo	AppInfo is invalid.
709	No such Application is running	Any of applications' <u><runningStatus></u> s is not " <u>Running</u> ".
710	Not stoppable application	Any of applications listed by the <u>@ids</u> in the <u>AppIDs</u> cannot be stopped.
711	Application is installed	Application's <u><installationStatus></u> is already " <u>Installed</u> ".
712	No such Application is installed	Any of applications' <u><installationStatus></u> s is not " <u>Installed</u> ".

errorCode	errorDescription	Description
713	Not uninstallable application	Any of applications listed by the <i>@ids</i> in the <i>AppIDs</i> cannot be uninstalled.
714	Invalid ConnectionID	One ore more of the connection IDs in the <i>ConnectionIDs</i> are not valid.
715	Not enough storage	Storage is not enough to install the application.
716	Exceeding download limit	The application size exceeds the download limit.
717	Installation required	The application is required to be installed before it can be started.

Note: The errorDescription field returned by an action does not necessarily contain human-readable text (for example, as indicated in the second column of the Error Code tables). It can contain machine-readable information that provides more detailed information about the error. It is therefore not advisable for a control point to blindly display the errorDescription field contents to the user.

Note that 800-899 Error Codes are not permitted for standard actions. See subclause 3.3.2 of the UPnP Device Architecture specification [1] for more details.

6 XML Service Description

```
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <actionList>
    <action>
      <name>GetFeatureList</name>
      <argumentList>
        <argument>
          <name>FeatureList</name>
          <direction>out</direction>
          <relatedStateVariable>
            FeatureList
          </relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>GetAppInfoByIDs</name>
      <argumentList>
        <argument>
          <name>AppIDs</name>
          <direction>in</direction>
          <relatedStateVariable>
            A ARG TYPE AppIDs
          </relatedStateVariable>
        </argument>
        <argument>
          <name>AppInfo</name>
          <direction>out</direction>
          <relatedStateVariable>
            A ARG TYPE AppInfo
          </relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>GetSupportedTargetFields</name>
      <argumentList>
        <argument>
          <name>SupportedTargetFields</name>
          <direction>out</direction>
        </argument>
      </argumentList>
    </action>
  </actionList>
</scpd>
```

```

        <relatedStateVariable>
            SupportedTargetFields
        </relatedStateVariable>
    </argument>
</argumentList>
</action>
<action>
    <name>GetAppIDList</name>
    <argumentList>
        <argument>
            <name>Target</name>
            <direction>in</direction>
            <relatedStateVariable>
                A ARG TYPE Target
            </relatedStateVariable>
        </argument>
        <argument>
            <name>TargetFields</name>
            <direction>in</direction>
            <relatedStateVariable>
                A ARG TYPE TargetFields
            </relatedStateVariable>
        </argument>
        <argument>
            <name>AppIDs</name>
            <direction>out</direction>
            <relatedStateVariable>
                A ARG TYPE AppIDs
            </relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>GetRunningAppList</name>
    <argumentList>
        <argument>
            <name>RunningAppList</name>
            <direction>out</direction>
            <relatedStateVariable>
                RunningAppList
            </relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>GetRunningStatus</name>
    <argumentList>
        <argument>
            <name>AppIDs</name>
            <direction>in</direction>
            <relatedStateVariable>
                A ARG TYPE AppIDs
            </relatedStateVariable>
        </argument>
        <argument>
            <name>RunningStatus</name>
            <direction>out</direction>
            <relatedStateVariable>
                A ARG TYPE AppInfo
            </relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>StartAppByID</name>
    <argumentList>
        <argument>
            <name>AppID</name>
            <direction>in</direction>

```



```

        <relatedStateVariable>
            A ARG TYPE AppIDs
        </relatedStateVariable>
    </argument>
</argument>
<argument>
    <name>StartParameters</name>
    <direction>in</direction>
    <relatedStateVariable>
        A ARG TYPE Parameters
    </relatedStateVariable>
</argument>
</argumentList>
</action>
<action>
    <name>StartAppByURI</name>
    <argumentList>
        <argument>
            <name>StartURI</name>
            <direction>in</direction>
            <relatedStateVariable>
                A ARG TYPE URI
            </relatedStateVariable>
        </argument>
        <argument>
            <name>AppInfo</name>
            <direction>in</direction>
            <relatedStateVariable>
                A ARG TYPE AppInfo
            </relatedStateVariable>
        </argument>
        <argument>
            <name>StartParameters</name>
            <direction>in</direction>
            <relatedStateVariable>
                A ARG TYPE Parameters
            </relatedStateVariable>
        </argument>
        <argument>
            <name>AppID</name>
            <direction>out</direction>
            <relatedStateVariable>
                A ARG TYPE AppIDs
            </relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>StopApp</name>
    <argumentList>
        <argument>
            <name>AppIDs</name>
            <direction>in</direction>
            <relatedStateVariable>
                A ARG TYPE AppIDs
            </relatedStateVariable>
        </argument>
        <argument>
            <name>StoppedAppIDs</name>
            <direction>out</direction>
            <relatedStateVariable>
                A ARG TYPE AppIDs
            </relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>InstallAppByID</name>
    <argumentList>
        <argument>

```

```

        <name>AppID</name>
        <direction>in</direction>
        <relatedStateVariable>
            A ARG TYPE AppIDs
        </relatedStateVariable>
    </argument>
</argumentList>
</action>
<action>
    <name>InstallAppByURI</name>
    <argumentList>
        <argument>
            <name>InstallationURI</name>
            <direction>in</direction>
            <relatedStateVariable>
                A ARG TYPE URI
            </relatedStateVariable>
        </argument>
        <argument>
            <name>AppInfo</name>
            <direction>in</direction>
            <relatedStateVariable>
                A ARG TYPE AppInfo
            </relatedStateVariable>
        </argument>
        <argument>
            <name>InstallParameters</name>
            <direction>in</direction>
            <relatedStateVariable>
                A ARG TYPE Parameters
            </relatedStateVariable>
        </argument>
        <argument>
            <name>AppID</name>
            <direction>out</direction>
            <relatedStateVariable>
                A ARG TYPE AppIDs
            </relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>UninstallApp</name>
    <argumentList>
        <argument>
            <name>AppIDs</name>
            <direction>in</direction>
            <relatedStateVariable>
                A ARG TYPE AppIDs
            </relatedStateVariable>
        </argument>
        <argument>
            <name>UninstalledAppIDs</name>
            <direction>out</direction>
            <relatedStateVariable>
                A ARG TYPE AppIDs
            </relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>GetInstallationStatus</name>

```

```

<argumentList>
  <argument>
    <name>AppIDs</name>
    <direction>in</direction>
    <relatedStateVariable>
      A ARG TYPE AppIDs
    </relatedStateVariable>
  </argument>
  <argument>
    <name>InstallationStatus</name>
    <direction>out</direction>
    <relatedStateVariable>
      A ARG TYPE AppInfo
    </relatedStateVariable>
  </argument>
</argumentList>
</action>
<action>
  <name>GetAppConnectionInfo</name>
  <argumentList>
    <argument>
      <name>AppIDs</name>
      <direction>in</direction>
      <relatedStateVariable>
        A ARG TYPE AppIDs
      </relatedStateVariable>
    </argument>
    <argument>
      <name>ConnectionInfo</name>
      <direction>out</direction>
      <relatedStateVariable>
        A ARG TYPE AppInfo
      </relatedStateVariable>
    </argument>
  </argumentList>
</action>
<action>
  <name>ConnectApptoApp</name>
  <argumentList>
    <argument>
      <name>AppID</name>
      <direction>in</direction>
      <relatedStateVariable>
        A ARG TYPE AppIDs
      </relatedStateVariable>
    </argument>
    <argument>
      <name>ConnectionID</name>
      <direction>out</direction>
      <relatedStateVariable>
        A ARG TYPE ConnectionIDs
      </relatedStateVariable>
    </argument>
  </argumentList>
</action>
<action>
  <name>DisconnectApptoApp</name>
  <argumentList>
    <argument>
      <name>ConnectionIDs</name>
      <direction>in</direction>
      <relatedStateVariable>
        A ARG TYPE ConnectionIDs
      </relatedStateVariable>
    </argument>
    <argument>
      <name>DisconnectedConnectionIDs</name>
      <direction>out</direction>
      <relatedStateVariable>

```

```

        A ARG TYPE ConnectionIDs
    </relatedStateVariable>
</argument>
</argumentList>
</action>
<action>
    <name>GetCurrentConnectionInfo</name>
    <argumentList>
        <argument>
            <name>ConnectionIDs</name>
            <direction>out</direction>
            <relatedStateVariable>
                A ARG TYPE ConnectionIDs
            </relatedStateVariable>
        </argument>
        <argument>
            <name>ConnectionAppIDs</name>
            <direction>out</direction>
            <relatedStateVariable>
                A ARG TYPE AppIDs
            </relatedStateVariable>
        </argument>
    </argumentList>
</action>
<!--Declarations for other actions added by UPnP vendor
(if any) go here-->
</actionList>
<serviceStateTable>
    <stateVariable sendEvents="no">
        <name>FeatureList</name>
        <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
        <name>A ARG TYPE AppIDs</name>
        <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
        <name>AppInfoList</name>
        <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
        <name>A ARG TYPE AppInfo</name>
        <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
        <name>SupportedTargetFields</name>
        <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
        <name>A ARG TYPE Target</name>
        <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
        <name>A ARG TYPE TargetFields</name>
        <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="yes">
        <name>RunningAppList</name>
        <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="yes">
        <name>TransitioningApps</name>
        <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
        <name>A ARG TYPE URI</name>
        <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">

```

```
<name>A_ARG_TYPE_Parameters</name>
<dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>A_ARG_TYPE_ConnectionIDs</name>
  <dataType>string</dataType>
</stateVariable>
<!--Declarations for other state variables added by
UPnP vendor (if any) go here-->
</serviceStateTable>
</scpd>
```

7 Test

No semantic tests have been specified for this service.