ApplicationManagement:1 Service

For UPnP Version 1.0
Status: Standardized DCP (SDCP)
Date: September 30, 2014
Service Template Version 3.0

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

This document specifies the characteristics of the UPnP networked service named *ApplicationManagement*, version *1*. 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:1*, UPnP Forum, September 30, 2014.
Available at: http://www.upnp.org/specs/ms/UPnP-ms-ScreenDevice-v1-Device-20140930.pdf.

Latest version available at: http://www.upnp.org/specs/ms/UPnP-ms-ScreenDevice-v1-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 AppInfoList XML Structures*, UPnP Forum, September 30, 2014. Available at: http://www.upnp.org/ schemas/ms/AppInfoList-v1-20140930.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 *SECURITY feature*

An extension of the DeviceProtection service [9] to the actions (*Action Level Access*) of the ApplicationManagement service (see subclause 5.2.1).

3.1.2 *AM Roles*

Access level of a *Control Point* or *User Identity* to authorize a specific set of the ApplicationManagement actions (see subclauses 5.2.1.1 and 5.2.1.2).

3.1.3 Application

A software program designed to help people perform an activity.

3.1.4 Native Application

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

3.1.5 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:1*

5.2 Key Concepts

5.2.1 *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 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 1 — 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

5.2.1.1 *AM* (ApplicationManagement) *Roles*

The following Table 2 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 2 — ApplicationManagement *Roles*

Role Name	R/A
Public	*CR*
Basic	*CR*
Admin	*CR*
Conditionally required if the *SECURITY feature* is supported, 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 3 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 3 for details).

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.

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

ApplicationManagement actions are defined as *Restrictable* or *Non-Restrictable* (see Table 3) only when the *SECURITY feature* is supported.

The Table 3 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 3 — Action to *Role* Permission Mapping

Action Name	Category	Role	
		Public	***Basic***
GetAppInfoByIDs()	*Restrictable*	*NO*	*YES*
GetSupportedTargetFields()	*Non-Restrictable*	*YES*	*YES*
GetAppIDList()	*Restrictable*	*NO*	*YES*
GetRunningAppList()	*Restrictable*	*NO*	*YES*
GetRunningStatus()	*Restrictable*	*NO*	*YES*
StartAppByID()	*Restrictable*	*NO*	*YES*
StartAppByURI()	*Restrictable*	*NO*	*YES*
StopApp()	*Restrictable*	*NO*	*YES*
GetAppConnectionInfo()	*Restrictable*	*NO*	*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 4 —State variables

Variable Name	R/A [a]	Data Type	Allowed Value	Default Value	Eng. Units
A_ARG_TYPE_AppIDs	*R*	**string**	CSV(**string**) See subclause 5.3.1		
AppInfoList	*R*	**string**	*AppInfoList XML Document* See subclause 5.3.2		
A_ARG_TYPE_AppInfo	*R*	**string**	*XML fragment of AppInfoList XML Document* See subclause 5.3.3		
SupportedTargetFields	*R*	**string**	CSV(**string**) See subclause 5.3.4		
A_ARG_TYPE_Target	*R*	**string**	See subclause 5.3.5		
A_ARG_TYPE_TargetFields	*R*	**string**	CSV(**string**) See subclause 5.3.6		
RunningAppList	*R*	**string**	CSV(**string**) See subclause 5.3.7		
A_ARG_TYPE_URI	*A*	**string**	See subclause 5.3.8		
A_ARG_TYPE_Parameters	*R*	**string**	See subclause 5.3.9		
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.

5.3.1 *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 5.3.2).

5.3.2 *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 [10] 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>
      <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>
    </application>
</AppInfoList>
```

<xml>
Allowed. Case sensitive.

<AppInfoList>
Required. <XML>. Shall include a namespace declaration for the XML Schema for AppInfoList XML Structures [10] ("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>
Required. 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*".

<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>
Allowed. <XML>. 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.

Table 5 — 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 6 — 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>
```

5.3.3 *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 5.3.2). It shall contain zero or more <*application*> element(s), its (their) attributes and sub-elements which depend on the invoked actions.

5.3.4 *SupportedTargetFields*

This required state variable provides an unordered CSV list of the searchable fields by the *GetAppIDList()* action (see subclause 5.5.3), 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 7. The required fields can be expanded in future versions of the specification.

Table 7 — Required fields for *SupportedTargetFields*

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

5.3.5 *A_ARG_TYPE_Target*

This required state variable provides type information the *Target* input argument in the *GetAppIDList()* action (see subclause 5.5.3)

5.3.6 *A_ARG_TYPE_TargetFields*

This required state variable provides type information for the *TargetFields* input argument in the *GetAppIDList()* action (see subclause 5.5.3). This state variable is an unordered CSV list of the values in the CSV list of the *SupportedTargetFields* state variable.

5.3.7 *RunningAppList*

This required 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 5.3.2). This state variable shall have an empty string when there are no applications with the <*runningStatus*> element set to "*Running*".

5.3.8 *A_ARG_TYPE_URI*

This allowed state variable provides type information for the *StartURI* input argument in the *StartAppByURI()* action (see subclause 5.5.7). This state variable shall be properly escaped as described in [4]. In addition, it shall be escaped according to the requirements in [5].

5.3.9 *A_ARG_TYPE_Parameters*

This required state variable provides type information for the *StartParameters* input arguments in various actions (see subclauses 5.5.6 and 5.5.7). The arguments are used for the according actions to be successfully accepted, and the proper values are application-specific.

5.4 Eventing and Moderation

Table 8 — Event moderation

Variable Name	Evented	Moderated Event	Min Event Interval [a] (seconds)	Logical Combination	Min Delta per Event [b]
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		
A_ARG_TYPE_URI	*NO*	*NO*			
A_ARG_TYPE_Parameters	*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 9 — Actions

Name	R/A [a]	Control Point R/A [b]
GetAppInfoByIDs()	R	R
GetSupportedTargetFields()	R	A
GetAppIDList()	R	R
GetRunningAppList()	R	A
GetRunningStatus()	R	A
StartAppByID()	R	A
StartAppByURI()	A	A
StopApp()	A	A
GetAppConnectionInfo()	A	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 See action description for conditions under which implementation of this action is required.

d See action description for conditions under which implementation of this action is allowed. If the condition is not met implementation of this action is not allowed.

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 *GetAppInfoByIDs()*

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

5.5.1.1 Arguments

- *AppIDs*: Specifies applications to retrieve their information. See subclause 5.3.1. 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 5.3.2 and 5.3.3). 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 10 — Arguments for *GetAppInfoByIDs()*

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

5.5.1.2 Dependency on State

None.

5.5.1.3 Effect on State

None.

5.5.1.4 Errors

Table 11 — 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*.

5.5.2 *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 5.5.3) on the Screen Device.

5.5.2.1 Arguments

Table 12 — Arguments for *GetSupportedTargetFields()*

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

5.5.2.2 Dependency on State

None.

5.5.2.3 Effect on State

None.

5.5.2.4 Errors

Table 13 — 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].

5.5.3 *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 5.3.5) 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 5.3.4 and 5.3.6).

5.5.3.1 Arguments

Table 14 — 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*

5.5.3.2 Dependency on State

None.

5.5.3.3 Effect on State

None.

5.5.3.4 Errors

Table 15 — 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].
703	Invalid TargetFields	*TargetFields* contains unsupported values.

5.5.4 *GetRunningAppList()*

This required 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 5.3.7).

5.5.4.1 Arguments

Table 16 — Arguments for *GetRunningAppList()*

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

5.5.4.2 Dependency on State

None.

5.5.4.3 Effect on State

None.

5.5.4.4 Errors

Table 17 — 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].

5.5.5 *GetRunningStatus()*

This required action enables a Screen Control Point to retrieve the running status of applications specified by the *AppIDs* argument.

5.5.5.1 Arguments

- *AppIDs*: Specifies applications to retrieve their running status. See subclause 5.3.1.
- *RunningStatus*: an XML fragment of the *AppInfoList* state variable (see subclauses 5.3.2 and 5.3.3). 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 18 — Arguments for *GetRunningStatus()*

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

5.5.5.2 Dependency on State

None.

5.5.5.3 Effect on State

None.

5.5.5.4 Errors

Table 19 — 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*.

5.5.6 *StartAppByID()*

This required 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*".

5.5.6.1 Arguments

- *AppID*: Specifies the application to be started. See subclause 5.3.1. This argument shall contain only a single *application@id* value.
- *StartParameters*: see subclause 5.3.9.

Table 20 — Arguments for *StartAppByID()*

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

5.5.6.2 Dependency on State

None.

5.5.6.3 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*".

5.5.6.4 Errors

Table 21 — 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.

5.5.7 *StartAppbyURI()*

This allowed 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*".

5.5.7.1 Arguments

- *StartURI*: Provides the <*startURI*> to start an application. See subclause 5.3.8.
- *AppInfo*: an XML fragment of the *AppInfoList* state variable (see subclauses 5.3.2 and 5.3.3). Provides the additional information for the application to be started. The <*friendlyName*> is required.
- *StartParameters*: see subclause 5.3.9.
- *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 22 — 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*

5.5.7.2 Dependency on State

None.

5.5.7.3 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*".

5.5.7.4 Errors

Table 23 — 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.

5.5.8 *StopApp()*

This allowed action stops applications specified by the *AppIDs* input argument on the Screen Device when successfully accepted.

5.5.8.1 Arguments

- *AppIDs*: Specifies the applications to be stopped. See subclause 5.3.1.
- *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 24 — Arguments for *StopApp()*

Argument	Direction	Related State Variable
AppIDs	*IN*	*A_ARG_TYPE_AppIDs*
StoppedAppIDs	*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 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*".

5.5.8.4 Errors

Table 25 — 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.

5.5.9 *GetAppConnectionInfo()*

This allowed action enables a Screen Control Point to retrieve the app-to-app connection information of applications specified by the *AppIDs* input argument.

5.5.9.1 Arguments

- *AppIDs*: Specifies the application to retrieve their app-to-app connection information. See subclause 5.3.1.
- *ConnectionInfo*: an XML fragment of the *AppInfoList* state variable (see subclauses 5.3.2 and 5.3.3). 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 26 — Arguments for *GetAppConnectionInfo()*

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

5.5.9.2 Dependency on State

None.

5.5.9.3 Effect on State

None.

5.5.9.4 Errors

Table 27 — 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*".

5.5.10 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]).

5.5.11 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 28 — 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 *@id*s in the *AppIDs* are not valid.
702	Too many IDs	Too many *@id*s specified in the *AppIDs*.
703	Invalid TargetFields	*TargetFields* 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.

errorCode	errorDescription	Description
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.
712	No such Application is installed	Any of applications' <*installationStatus*>s is not "*Installed*".
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>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>
               <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>
            <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>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>
        <!--Declarations for other actions added by UPnP vendor
        (if any) go here-->
    </actionList>
    <serviceStateTable>
        <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="no">
            <name>A_ARG_TYPE_URI</name>
            <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_Parameters</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.