**Application Management: 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.

<table>
<thead>
<tr>
<th>Authors</th>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clarke Stevens</td>
<td>CableLabs</td>
</tr>
<tr>
<td>Wouter van der Beek</td>
<td>Cisco Systems Inc.</td>
</tr>
<tr>
<td>Seung R. Yang (Chair)</td>
<td>LG Electronics</td>
</tr>
<tr>
<td>Anders Klemets</td>
<td>Microsoft</td>
</tr>
<tr>
<td>Nicholas Frame</td>
<td>TP Vision</td>
</tr>
</tbody>
</table>

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.
## 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 FeatureList .................................................................................................... 9
      5.3.2 A_ARG_TYPE_AppIDs ............................................................................... 9
      5.3.3 ApplInfoList ................................................................................................ 10
      5.3.4 A_ARG_TYPE_ApplInfo ............................................................................. 13
      5.3.5 SupportedTargetFields .............................................................................. 14
      5.3.6 A_ARG_TYPE_Target ............................................................................... 14
      5.3.7 A_ARG_TYPE_TargetFields ...................................................................... 14
      5.3.8 RunningAppList .......................................................................................... 14
      5.3.9 TransitioningApps ....................................................................................... 14
      5.3.10 A_ARG_TYPE_URI ................................................................................ 14
      5.3.11 A_ARG_TYPE_Parameters ...................................................................... 14
      5.3.12 A_ARG_TYPE_ConnectionIDs ................................................................. 15
   5.4 Eventing and Moderation .......................................................................................... 15
   5.5 Actions ..................................................................................................................... 15
      5.5.1 GetFeatureList() ......................................................................................... 16
      5.5.2 GetAppInfoByIDs() ................................................................................... 17
      5.5.3 GetSupportedTargetFields() ..................................................................... 17
      5.5.4 GetIDL() ..................................................................................................... 18
      5.5.5 GetRunningAppList() ................................................................................ 19
      5.5.6 GetRunningStatus() ................................................................................... 19
      5.5.7 StartAppByID() ......................................................................................... 20
      5.5.8 StartAppbyURI() ....................................................................................... 21
      5.5.9 StopApp() ................................................................................................. 22
      5.5.10 InstallAppByID() ..................................................................................... 23
      5.5.11 InstallAppByURI() ................................................................................... 24
      5.5.12 UninstallApp() .......................................................................................... 25
      5.5.13 GetInstallationStatus() ......................................................................... 26
      5.5.14 GetAppConnectionInfo() ......................................................................... 27
      5.5.15 ConnectApptoApp() ................................................................................ 28
      5.5.16 DisconnectApptoApp() .......................................................................... 28
      5.5.17 GetCurrentConnectionInfo() ................................................................... 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 — AMS features ........................................................................................................ 6
Table 2 — Error Codes for Action Level Access ........................................................................7
Table 3 — ApplicationManagement Roles .............................................................................. 7
Table 4 — Action to Role Permission Mapping ...................................................................... 8
Table 5 — State variables ....................................................................................................... 9
Table 6 — Allowed values for function element .................................................................... 12
Table 7 — Allowed values for connectionAddress element .................................................. 12
Table 8 — Required fields for SupportedTargetFields ........................................................ 14
Table 9 — Event moderation .................................................................................................. 15
Table 10 — Actions ................................................................................................................ 16
Table 11 — Arguments for GetFeatureList() ....................................................................... 16
Table 12 — Error Codes for GetFeatureList() ...................................................................... 17
Table 13 — Arguments for GetAppInfoByIds() ..................................................................... 17
Table 14 — Error Codes for GetAppInfoByIds() ................................................................... 17
Table 15 — Arguments for GetSupportedTargetFields() ....................................................... 18
Table 16 — Error Codes for GetSupportedTargetFields() .................................................... 18
Table 17 — Arguments for GetAppIDList() .......................................................................... 18
Table 18 — Error Codes for GetAppIDList() ........................................................................ 18
Table 19 — Arguments for GetRunningAppList() .................................................................. 19
Table 20 — Error Codes for GetRunningAppList() ............................................................... 19
Table 21 — Arguments for GetRunningStatus() ................................................................... 19
Table 22 — Error Codes for GetRunningStatus() ................................................................ 20
Table 23 — Arguments for StartAppById() .......................................................................... 20
Table 24 — Error Codes for StartAppById() ........................................................................ 20
Table 25 — Arguments for StartAppByURI() ....................................................................... 21
Table 26 — Error Codes for StartAppByURI() ..................................................................... 21
Table 27 — Arguments for StopApp() .................................................................................. 22
Table 28 — Error Codes for StopApp() ............................................................................... 23
Table 29 — Arguments for InstallAppById() ....................................................................... 23
Table 30 — Error Codes for InstallAppById() ...................................................................... 24
Table 31 — Arguments for InstallAppByURI() .................................................................... 25
Table 32 — Error Codes for InstallAppByURI() ................................................................... 25
Table 33 — Arguments for UninstallApp() .......................................................................... 26
Table 34 — Error Codes for UninstallApp() ....................................................................... 26
Table 35 — Arguments for GetInstallationStatus() ............................................................... 27
Table 36 — Error Codes for GetInstallationStatus() ............................................................. 27
Table 37 — Arguments for GetAppConnectionInfo() ............................................................. 27
Table 38 — Error Codes for GetAppConnectionInfo() .......................................................... 28
Table 39 — Arguments for ConnectAppToApp() ................................................................. 28
Table 40 — Error Codes for ConnectAppToApp() ................................................................. 28
Table 41 — Arguments for DisconnectAppToApp() .............................................................. 29
Table 42 — Error Codes for DisconnectAppToApp() ............................................................. 29
Table 43 — Arguments for GetCurrentConnectionInfo() ...................................................... 29
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.


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:

\[\text{urn}:\text{schemas-upnp-org:service:ApplicationManagement:2}\]

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

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Version</th>
<th>Description</th>
<th>Requirements</th>
</tr>
</thead>
</table>
| Default (no feature) | | Default functionalities | GetAppInfoByID()  
GetSupportedTargetFieldList()  
GetAppIDList()  
and the related state variables |
| START | 1 | Application-starting related functionalities | StartAppByID()  
StartAppByURL()  
StopApp()  
GetRunningAppList()  
GetRunningStatus()  
and the related state variables |
| INSTALL | 1 | Application-installing related functionalities | InstallAppByID()  
InstallAppByURI()  
UninstallApp()  
GetInstallationStatus()  
and the related state variables |
| CONNECT | 1 | Application-connecting related functionalities | GetAppConnectionInfo()  
ConnectAppToApp()  
DisconnectAppToApp()  
GetCurrentConnectionAppInfo()  
and the related state variables |
| SECURITY | 1 | Action Level Access | See subclause 5.2.2. |

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

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

#### 5.2.2 SECURITY feature

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

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

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

Table 2 — Error Codes for Action Level Access

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

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

<table>
<thead>
<tr>
<th>Role Name</th>
<th>R/A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public</td>
<td>CR</td>
</tr>
<tr>
<td>Basic</td>
<td>CR</td>
</tr>
<tr>
<td>AM_SuperUser</td>
<td>CR</td>
</tr>
<tr>
<td>Admin</td>
<td>CR</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Conditionally required if the SECURITY feature is implemented, and not allowed otherwise.</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Action Name</th>
<th>Category</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Public</td>
</tr>
<tr>
<td>GetFeatureList()</td>
<td>Non-Restrictable</td>
<td>YES</td>
</tr>
<tr>
<td>GetAppInfoByIds()</td>
<td>Restrictable</td>
<td>NO</td>
</tr>
<tr>
<td>GetSupportedTargetFields()</td>
<td>Non-Restrictable</td>
<td>YES</td>
</tr>
<tr>
<td>GetAppIdList()</td>
<td>Restrictable</td>
<td>NO</td>
</tr>
<tr>
<td>GetRunningAppList()</td>
<td>Restrictable</td>
<td>NO</td>
</tr>
<tr>
<td>GetRunningStatus()</td>
<td>Restrictable</td>
<td>NO</td>
</tr>
<tr>
<td>StartAppById()</td>
<td>Restrictable</td>
<td>NO</td>
</tr>
<tr>
<td>StartAppByUri()</td>
<td>Restrictable</td>
<td>NO</td>
</tr>
<tr>
<td>StopApp()</td>
<td>Restrictable</td>
<td>NO</td>
</tr>
<tr>
<td>InstallAppById()</td>
<td>Restrictable</td>
<td>NO</td>
</tr>
<tr>
<td>InstallAppByUri()</td>
<td>Restrictable</td>
<td>NO</td>
</tr>
<tr>
<td>UninstallApp()</td>
<td>Restrictable</td>
<td>NO</td>
</tr>
<tr>
<td>GetInstallationStatus()</td>
<td>Restrictable</td>
<td>NO</td>
</tr>
<tr>
<td>GetAppConnectionInfo()</td>
<td>Restrictable</td>
<td>NO</td>
</tr>
<tr>
<td>ConnectAppToApp()</td>
<td>Restrictable</td>
<td>NO</td>
</tr>
<tr>
<td>DisconnectAppToApp()</td>
<td>Restrictable</td>
<td>NO</td>
</tr>
<tr>
<td>GetCurrentConnectionInfo()</td>
<td>Restrictable</td>
<td>NO</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>R/A a</th>
<th>Data Type</th>
<th>Allowed Value</th>
<th>Default Value</th>
<th>Eng. Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>FeatureList</td>
<td>R</td>
<td>string</td>
<td>CSV(string)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_AppIDs</td>
<td>R</td>
<td>string</td>
<td>CSV(string)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AppInfoList</td>
<td>R</td>
<td>string</td>
<td>AppInfoList XML Document</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_AppInfo</td>
<td>R</td>
<td>string</td>
<td>XML fragment of AppInfoList XML Document</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SupportedTargetFields</td>
<td>R</td>
<td>string</td>
<td>CSV(string)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_Target</td>
<td>R</td>
<td>string</td>
<td>CSV(string)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_TargetFields</td>
<td>R</td>
<td>string</td>
<td>CSV(string)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RunningAppList</td>
<td>CR c</td>
<td>string</td>
<td>CSV(string)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TransitioningApps</td>
<td>CR c</td>
<td>string</td>
<td>CSV(string)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_URI</td>
<td>CR c</td>
<td>string</td>
<td>See subclause 5.3.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_Parameter</td>
<td>CR c</td>
<td>string</td>
<td>See subclause 5.3.11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_ConnectionIDs</td>
<td>CR b</td>
<td>string</td>
<td>CSV(string)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-standard state variables</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>implemented by a UPnP vendor go here</td>
<td>X</td>
<td>TBD</td>
<td>TBD</td>
<td>TBD</td>
<td>TBD</td>
</tr>
</tbody>
</table>

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
<?xml version="1.0" encoding="UTF-8"?>
<FeatureList>
  <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
<xml version="1.0" encoding="UTF-8">
<AppInfoList
    xmlns="urn:schemas-upnp-org:ms:AppInfoList"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    <application>
        <marketAppID>Unique Identifier of the Application</marketAppID>
        <version>Application Version</version>
        <friendlyName>Language of friendlyName</friendlyName>
        <alternativeID>Organization Name</alternativeID>
        <function>Organization Name</function>
        <runningStatus>Activation Status of Application</runningStatus>
        <startURI=deviceType>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>URI for Installation of the Application</installationURI>
        <usagePolicy>Restriction Information</usagePolicy>
        <apolloAppInfo>
            <matchingProtocolName>User-friendly protocol name</matchingProtocolName>
            <protocol>Protocol name</protocol>
            <connectionAddress>Information needed to establish a connection to the Application</connectionAddress>
        </apolloAppInfo>
        <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>**

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

  - **@id**
    Required. xsd:string. Provides 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 1 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", "Transiting", "Transiting_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 "Installed" 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".

© 2014, UPnP Forum. All rights Reserved.
<appenToAppInfo>
  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).
  </matchingProtocolName>

  <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.
  </protocol>

  <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 6 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.
  </connectionAddress>

  <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 subelements:

      <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).
      </mimetype>

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

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

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

      <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.
      </url>
  </iconList>

  Table 6 — Allowed values for function element

<table>
<thead>
<tr>
<th>@org</th>
<th>function_value</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>upnp.org</td>
<td>urn:schemas-upnp-org:device:deviceType:v</td>
<td>[1]</td>
</tr>
<tr>
<td>upnp.org</td>
<td>urn:schemas-upnp-org:service:serviceType:v</td>
<td>[1]</td>
</tr>
<tr>
<td>vendor-defined</td>
<td>vendor-defined</td>
<td></td>
</tr>
</tbody>
</table>

  Table 7 — Allowed values for connectionAddress element

<table>
<thead>
<tr>
<th>protocol</th>
<th>connectionAddress_value</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP</td>
<td>An absolute http or https URI</td>
<td>[4], [5]</td>
</tr>
<tr>
<td>Websocket</td>
<td>WebSocket URI</td>
<td>[6]</td>
</tr>
<tr>
<td>UPnP</td>
<td>uuid:device-uuid</td>
<td>[1]</td>
</tr>
<tr>
<td>vendor-defined</td>
<td>vendor-defined</td>
<td></td>
</tr>
</tbody>
</table>

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

© 2014, UPnP Forum. All rights Reserved.
Example:

<?xml version="1.0" encoding="UTF-8"?>
<AppInfoList
xmlns="urn:schemas-upnp-org:ms:AppInfoList"
xmllns: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-AB6B-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"
xmllns: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>
    <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>
<thead>
<tr>
<th>Value</th>
<th>Parameter in AppInfoList</th>
<th>R/A</th>
</tr>
</thead>
<tbody>
<tr>
<td>friendlyName</td>
<td>application::friendlyName</td>
<td>R</td>
</tr>
<tr>
<td>matchingProtocolName</td>
<td>application::appInfo::matchingProtocolName</td>
<td>R</td>
</tr>
</tbody>
</table>

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 subclauses 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

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Evented</th>
<th>Moderated Event</th>
<th>Min Event Interval a (seconds)</th>
<th>Logical Combination</th>
<th>Min Delta per Event b</th>
</tr>
</thead>
<tbody>
<tr>
<td>FeatureList</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_AppIDs</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AppInfoList</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_AppInfo</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SupportedTargetFields</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_Target</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_TargetFields</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RunningAppList</td>
<td>YES</td>
<td>YES</td>
<td>0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TransitioningApps</td>
<td>YES</td>
<td>YES</td>
<td>0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_URI</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_Parameters</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_ConnectionIDs</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 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 $(N) \times (allowedValueRange \ Step)$

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

<table>
<thead>
<tr>
<th>Name</th>
<th>R/A a</th>
<th>Control Point R/A b</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetFeatureList()</td>
<td>CR c</td>
<td>A</td>
</tr>
<tr>
<td>GetAppInfoByIDs()</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>GetSupportedTargetFields()</td>
<td>R</td>
<td>A</td>
</tr>
<tr>
<td>GetAppIDList()</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>GetRunningAppList()</td>
<td>CR d</td>
<td>A</td>
</tr>
<tr>
<td>GetRunningStatus()</td>
<td>CR d</td>
<td>A</td>
</tr>
<tr>
<td>StartAppByID()</td>
<td>CR d</td>
<td>A</td>
</tr>
<tr>
<td>StartAppByURL()</td>
<td>CR d</td>
<td>A</td>
</tr>
<tr>
<td>StopApp()</td>
<td>CR d</td>
<td>A</td>
</tr>
<tr>
<td>InstallAppByID()</td>
<td>CR c</td>
<td>A</td>
</tr>
<tr>
<td>InstallAppByURL()</td>
<td>CR c</td>
<td>A</td>
</tr>
<tr>
<td>UninstallApp()</td>
<td>CR c</td>
<td>A</td>
</tr>
<tr>
<td>GetInstallationStatus()</td>
<td>CR c</td>
<td>A</td>
</tr>
<tr>
<td>GetAppConnectionInfo()</td>
<td>CR c</td>
<td>A</td>
</tr>
<tr>
<td>ConnectAppToApp()</td>
<td>CR c</td>
<td>A</td>
</tr>
<tr>
<td>DisconnectAppToApp()</td>
<td>CR c</td>
<td>A</td>
</tr>
<tr>
<td>GetCurrentConnectionInfo()</td>
<td>CR c</td>
<td>A</td>
</tr>
</tbody>
</table>

Non-standard actions implemented by an UPnP vendor go here.  

**a** For a device this column indicates whether the action shall be implemented or not, where R = required, A = allowed, CR = conditionally required, CA = conditionally allowed, X = Non-standard, add -D when deprecated (e.g., R-D, A-D).

**b** For a control point this column indicates whether a control point shall be capable of invoking this action, where R = required, A = allowed, CR = conditionally required, CA = conditionally allowed, X = Non-standard, add -D when deprecated (e.g., R-D, A-D).

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

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

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

### 5.5.1 GetFeatureList()

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

#### 5.5.1.1 Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>Related State Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>FeatureList</td>
<td>OUT</td>
<td>FeatureList</td>
</tr>
</tbody>
</table>

#### 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()

<table>
<thead>
<tr>
<th>errorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
</tbody>
</table>

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()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>Related State Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppIDs</td>
<td>IN</td>
<td>A_ARG_TYPE_AppIDs</td>
</tr>
<tr>
<td>AppInfo</td>
<td>OUT</td>
<td>A_ARG_TYPE_AppInfo</td>
</tr>
</tbody>
</table>

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()

<table>
<thead>
<tr>
<th>errorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>701</td>
<td>Invalid ID</td>
<td>One or more of the @ids in the AppIDs are not valid.</td>
</tr>
<tr>
<td>702</td>
<td>Too many IDs</td>
<td>Too many @ids specified in the AppIDs.</td>
</tr>
</tbody>
</table>

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()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>Related State Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>SupportedTargetFields</td>
<td>OUT</td>
<td>SupportedTargetFields</td>
</tr>
</tbody>
</table>

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()

<table>
<thead>
<tr>
<th>errorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
</tbody>
</table>

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()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>Related State Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>IN</td>
<td>A_ARG_TYPE_Target</td>
</tr>
<tr>
<td>TargetFields</td>
<td>IN</td>
<td>A_ARG_TYPE_TargetFields</td>
</tr>
<tr>
<td>AppIDs</td>
<td>OUT</td>
<td>A_ARG_TYPE_AppIDs</td>
</tr>
</tbody>
</table>

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()

<table>
<thead>
<tr>
<th>errorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
</tbody>
</table>
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()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>Related State Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>RunningAppList</td>
<td>OUT</td>
<td>RunningAppList</td>
</tr>
</tbody>
</table>

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()

<table>
<thead>
<tr>
<th>errorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>703</td>
<td>Invalid TargetFields</td>
<td>TargetFields contains unsupported values.</td>
</tr>
</tbody>
</table>

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()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>Related State Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppIDs</td>
<td>IN</td>
<td>A_ARG_TYPE_AppIDs</td>
</tr>
<tr>
<td>RunningStatus</td>
<td>OUT</td>
<td>A_ARG_TYPE_AppInfo</td>
</tr>
</tbody>
</table>

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()

<table>
<thead>
<tr>
<th>errorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>701</td>
<td>Invalid ID</td>
<td>One or more of the @ids in the AppIDs are not valid.</td>
</tr>
<tr>
<td>702</td>
<td>Too many IDs</td>
<td>Too many @ids specified in the AppIDs.</td>
</tr>
</tbody>
</table>

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()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>Related State Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppID</td>
<td>IN</td>
<td>A_ARG_TYPE_AppIDs</td>
</tr>
<tr>
<td>StartParameters</td>
<td>IN</td>
<td>A_ARG_TYPE_Parameters</td>
</tr>
</tbody>
</table>

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()

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

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()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>Related State Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>StartURI</td>
<td>IN</td>
<td>A_ARG_TYPE_URI</td>
</tr>
<tr>
<td>AppInfo</td>
<td>IN</td>
<td>A_ARG_TYPE_AppInfo</td>
</tr>
<tr>
<td>StartParameters</td>
<td>IN</td>
<td>A_ARG_TYPE_Parameters</td>
</tr>
<tr>
<td>AppID</td>
<td>OUT</td>
<td>A_ARG_TYPE_AppIDs</td>
</tr>
</tbody>
</table>

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()

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

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()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>Related State Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppIDs</td>
<td>IN</td>
<td>A_ARG_TYPE_AppIDs</td>
</tr>
<tr>
<td>StoppedAppIDs</td>
<td>OUT</td>
<td>A_ARG_TYPE_StoppedAppIDs</td>
</tr>
</tbody>
</table>
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()

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Error Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>701</td>
<td>Invalid ID</td>
<td>One or more of the @ids in the AppIDs are not valid.</td>
</tr>
<tr>
<td>706</td>
<td>Rejected</td>
<td>This request is rejected, e.g., by Screen Device implementation or end user.</td>
</tr>
<tr>
<td>709</td>
<td>No such Application is running</td>
<td>Any of applications’ &lt;runningStatus&gt;s is not “Running”.</td>
</tr>
<tr>
<td>710</td>
<td>Not stoppable application</td>
<td>Any of applications listed by the @ids in the AppIDs cannot be stopped.</td>
</tr>
</tbody>
</table>

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()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>Related State Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppID</td>
<td>IN</td>
<td>A_ARG_TYPE_AppIDs</td>
</tr>
<tr>
<td>InstallParameters</td>
<td>IN</td>
<td>A_ARG_TYPE_Parameters</td>
</tr>
</tbody>
</table>

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

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()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>Related State Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>InstallationURI</td>
<td>IN</td>
<td>A_ARG_TYPE_URI</td>
</tr>
<tr>
<td>AppInfo</td>
<td>IN</td>
<td>A_ARG_TYPE_AppInfo</td>
</tr>
<tr>
<td>InstallParameters</td>
<td>IN</td>
<td>A_ARG_TYPE_Parameters</td>
</tr>
<tr>
<td>AppID</td>
<td>OUT</td>
<td>A_ARG_TYPE_AppIDs</td>
</tr>
</tbody>
</table>

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()

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

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>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>Related State Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppIDs</td>
<td>IN</td>
<td>A_ARG_TYPE_AppIDs</td>
</tr>
<tr>
<td>UninstalledAppIDs</td>
<td>OUT</td>
<td>A_ARG_TYPE_AppIDs</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>errorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>701</td>
<td>Invalid ID</td>
<td>One or more of the @ids in the AppIDs are not valid.</td>
</tr>
<tr>
<td>706</td>
<td>Rejected</td>
<td>This request is rejected, e.g., by Screen Device implementation or end user.</td>
</tr>
<tr>
<td>712</td>
<td>No such Application is installed</td>
<td>Any of applications’ &lt;installationStatus&gt;s is not “Installed”.</td>
</tr>
<tr>
<td>713</td>
<td>Not uninstallable application</td>
<td>Any of applications listed by the @ids in the AppIDs cannot be uninstalled.</td>
</tr>
</tbody>
</table>

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.

© 2014, UPnP Forum. All rights Reserved.
Table 35 — Arguments for `GetInstallationStatus()`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>Related State Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppIDs</td>
<td>IN</td>
<td>A_ARG_TYPE_AppIDs</td>
</tr>
<tr>
<td>InstallationStatus</td>
<td>OUT</td>
<td>A_ARG_TYPE_AppInfo</td>
</tr>
</tbody>
</table>

5.5.13.2 Dependency on State
None.

5.5.13.3 Effect on State
None.

5.5.13.4 Errors

<table>
<thead>
<tr>
<th>errorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>701</td>
<td>Invalid ID</td>
<td>One or more of the @id's in the AppIDs are not valid.</td>
</tr>
<tr>
<td>702</td>
<td>Too many IDs</td>
<td>Too many @id's specified in the AppIDs.</td>
</tr>
</tbody>
</table>

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 `<appptoAppInfo>` sub-elements if supported. If the `<appptoAppInfo>` 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()`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>Related State Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppIDs</td>
<td>IN</td>
<td>A_ARG_TYPE_AppIDs</td>
</tr>
<tr>
<td>ConnectionInfo</td>
<td>OUT</td>
<td>A_ARG_TYPE_AppInfo</td>
</tr>
</tbody>
</table>

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()

<table>
<thead>
<tr>
<th>errorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>701</td>
<td>Invalid ID</td>
<td>One or more of the @ids in the AppIDs are not valid.</td>
</tr>
<tr>
<td>709</td>
<td>No such Application is running</td>
<td>Any of applications’ &lt;runningStatus&gt;s is not “Running”.</td>
</tr>
</tbody>
</table>

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()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>Related State Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppID</td>
<td>IN</td>
<td>A_ARG_TYPE_AppIDs</td>
</tr>
<tr>
<td>ConnectionID</td>
<td>OUT</td>
<td>A_ARG_TYPE_ConnectionIDs</td>
</tr>
</tbody>
</table>

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()

<table>
<thead>
<tr>
<th>errorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>701</td>
<td>Invalid ID</td>
<td>One or more of the @ids in the AppIDs are not valid.</td>
</tr>
<tr>
<td>702</td>
<td>Too many IDs</td>
<td>Too many @ids specified in the AppIDs, i.e. more than one.</td>
</tr>
</tbody>
</table>

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()`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>Related State Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionIDs</td>
<td>IN</td>
<td>A_ARG_TYPE_ConnectionIDs</td>
</tr>
<tr>
<td>DisconnectedConnectionIDs</td>
<td>OUT</td>
<td>A_ARG_TYPE_ConnectionIDs</td>
</tr>
</tbody>
</table>

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()`

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

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()`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>Related State Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionIDs</td>
<td>OUT</td>
<td>A_ARG_TYPE_ConnectionIDs</td>
</tr>
</tbody>
</table>

© 2014, UPnP Forum. All rights Reserved.
### 5.5.17 Argument Direction

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>Related State Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionAppIDs</td>
<td>OUT</td>
<td>A_ARG_TYPE_AppIDs</td>
</tr>
</tbody>
</table>

#### 5.5.17.2 Dependency on State
None.

#### 5.5.17.3 Effect on State
None.

#### 5.5.17.4 Errors

<table>
<thead>
<tr>
<th>errorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
</tbody>
</table>

#### 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>
<thead>
<tr>
<th>errorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See clause 3 in the UPnP Device Architecture [1].</td>
</tr>
<tr>
<td>701</td>
<td>Invalid ID</td>
<td>One or more of the @ids in the AppIDs are not valid.</td>
</tr>
<tr>
<td>702</td>
<td>Too many IDs</td>
<td>Too many @ids specified in the AppIDs.</td>
</tr>
<tr>
<td>703</td>
<td>Invalid TargetFields</td>
<td>TargetFields contains unsupported values.</td>
</tr>
<tr>
<td>704</td>
<td>Invalid Parameter</td>
<td>The specified parameter is invalid</td>
</tr>
<tr>
<td>705</td>
<td>Application is running</td>
<td>Application is already running.</td>
</tr>
<tr>
<td>706</td>
<td>Rejected</td>
<td>This request is rejected, e.g., by Screen Device implementation or end user.</td>
</tr>
<tr>
<td>707</td>
<td>Invalid URI</td>
<td>The specified URL is invalid.</td>
</tr>
<tr>
<td>708</td>
<td>Invalid AppInfo</td>
<td>AppInfo is invalid.</td>
</tr>
<tr>
<td>709</td>
<td>No such Application is running</td>
<td>Any of applications’ &lt;runningStatus&gt;s is not “Running”.</td>
</tr>
<tr>
<td>710</td>
<td>Not stoppable application</td>
<td>Any of applications listed by the @ids in the AppIDs cannot be stopped.</td>
</tr>
<tr>
<td>711</td>
<td>Application is installed</td>
<td>Application’s &lt;installationStatus&gt; is already “Installed”.</td>
</tr>
<tr>
<td>712</td>
<td>No such Application is installed</td>
<td>Any of applications’ &lt;installationStatus&gt;s is not “Installed”.</td>
</tr>
<tr>
<td>errorCode</td>
<td>errorDescription</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>713</td>
<td>Not uninstallable application</td>
<td>Any of applications listed by the @ids in the AppIDs cannot be uninstalled.</td>
</tr>
<tr>
<td>714</td>
<td>Invalid ConnectionID</td>
<td>One or more of the connection IDs in the ConnectionIDs are not valid.</td>
</tr>
<tr>
<td>715</td>
<td>Not enough storage</td>
<td>Storage is not enough to install the application.</td>
</tr>
<tr>
<td>716</td>
<td>Exceeding download limit</td>
<td>The application size exceeds the download limit.</td>
</tr>
<tr>
<td>717</td>
<td>Installation required</td>
<td>The application is required to be installed before it can be started.</td>
</tr>
</tbody>
</table>

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
<?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>
          <relatedStateVariable>
            FeatureList
          </relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>GetAppInfoByIDs</name>
      <argumentList>
        <argument>
          <name>AppIDs</name>
          <relatedStateVariable>
            A_ARG_TYPE_AppIDs
          </relatedStateVariable>
        </argument>
        <argument>
          <name>AppInfo</name>
          <relatedStateVariable>
            A_ARG_TYPE_AppInfo
          </relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>GetSupportedTargetFields</name>
      <argumentList>
        <argument>
          <name>SupportedTargetFields</name>
          <relatedStateVariable>
            
          </relatedStateVariable>
        </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>
    </argument>
  </argumentList>
</action>
<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>InstallAppByID</name>
  <argumentList>
    <argument>
      <name>AppID</name>
      <direction>out</direction>
      <relatedStateVariable>
        A_ARG_TYPE_AppIDs
      </relatedStateVariable>
    </argument>
  </argumentList>
</action>
<argument><name>AppID</name><direction>in</direction><relatedStateVariable>A ARG TYPE AppIDs</relatedStateVariable></argument><argument><name>InstallParameters</name><direction>in</direction><relatedStateVariable>A ARG TYPE Parameters</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>
<action>
  <name>GetCurrentConnectionInfo</name>
  <argumentList>
    <argument>
      <name>ConnectionIDs</name>
      <direction>out</direction>
      <relatedStateVariable>connectionids</relatedStateVariable>
    </argument>
    <argument>
      <name>ConnectionAppIDs</name>
      <direction>out</direction>
      <relatedStateVariable>appidlist</relatedStateVariable>
    </argument>
  </argumentList>
</action>

<!-- Declarations for other actions added by UPnP vendor (if any) go here -->

<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_URI</name>
    <dataType>string</dataType>
  </stateVariable>
</serviceStateTable>
<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.