RemoteUIServer:1 Service Template Version 1.01

For UPnP™ Version *1.0*
Status: *Standardized DCP*
Date: *September 2, 2004*

Authors	Company
Jan van Ee	Philips
Mark R. Walker	Intel Corporation

Contents

1. OVERVIEW AND SCOPE 4

2. SERVICE MODELING DEFINITIONS 5

2.1. SERVICETYPE 5
2.2. STATE VARIABLES 5
2.2.1. UIListingUpdate 5
2.2.2. A_ARG_TYPE_DeviceProfile 5
2.2.3. A_ARG_TYPE_URI 6
2.2.4. A_ARG_TYPE_CompatibleUIs 6
2.2.5. A_ARG_TYPE_String 6
2.2.6. A_ARG_TYPE_Int 6
2.3. EVENTING AND MODERATION 6
2.4. ACTIONS 6
2.4.1. GetCompatibleUIs 7
2.4.2. SetUILifetime 8
2.4.3. Non-Standard Actions Implemented by a UPnP Vendor 9
2.4.4. Relationships Between Actions 9
2.4.5. Common Error Codes 9

3. THEORY OF OPERATION 10

3.1. EXAMPLE VALUES OF STATE VARIABLES 10
3.1.1. A_ARG_TYPE_URI 10
3.1.2. A_ARG_TYPE_CompatibleUIs 10
3.1.3. A_ARG_TYPE_DeviceProfile 14
3.1.4. UIListingUpdate 15
3.1.5. UIFilter Examples 16

4. A_ARG_TYPE_COMPATIBLEUIS XSD SCHEMA 17

5. DEVICEPROFILE XSD SCHEMA 18

6. XML SERVICE DESCRIPTION 19

List of Tables

Table 1: Service State Variables....5

Table 2: Event moderation....6

Table 3: Actions....6

Table 4: Arguments for `GetCompatibleUIs()`....7

Table 5: Arguments for SetUILifetime`()`....8

Table 6: Common Error Codes....9

Table 7: shortName values for known remoting protocols....13

1. Overview and Scope

This service definition is compliant with the UPnP Device Architecture version *1.0*.

This service is required for all Remote UI server devices.
It is specified in: **urn:schemas-upnp-org:service:*RemoteUIServerDevice***

2. Service Modeling Definitions

2.1. ServiceType

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

urn:schemas-upnp-org:service:*RemoteUIServer:1*.

2.2. State Variables

Table 1: Service State Variables

Variable Name	Req. or Opt.[1]	Data Type	Allowed Value	Default Value [2]	Eng. Units
UIListingUpdate	O	string	Undefined	Empty string	N/A
A_ARG_TYPE_DeviceProfile	R	string	Undefined	Empty string	N/A
A_ARG_TYPE_CompatibleUIs	R	string	Undefined	Empty string	N/A
A_ARG_TYPE_String	R	string	Undefined	Empty string	N/A
A_ARG_TYPE_URI	O	string	Undefined	Empty string	N/A
A_ARG_TYPE_Int	O	int	Undefined	0	N/A
Non-standard state variables implemented by a UPnP vendor go here.	*X*	*TBD*	*TBD*	*TBD*	*TBD*

[1] R = Required, O = Optional, X = Non-standard.

[2] Default values listed in this column are required. To specify standard optional values or to delegate assignment of values to the vendor, you must reference a specific instance of an appropriate table below.

2.2.1. UIListingUpdate

The optional, evented *UILisitingUpdate* state variable informs control points of changes in the listing of user interfaces exposed by the Remote UI server. *UILisitingUpdate* is composed of a list of unique identifiers corresponding to user interfaces that have changed since the last event. Each line in the listing, including the last line, must be terminated with a carriage-return character and a linefeed character. *UILisitingUpdate* values may consist of an empty string when the event is issued by the Remote UI server for the first time. An example value of *UILisitingUpdate* is shown in section 3.1.4.

2.2.2. A_ARG_TYPE_DeviceProfile

A_ARG_TYPE_DeviceProfile values are UTF-8 XML-formatted strings used by the UI client device to represent the list of all supported remoting protocols.

2.2.3. A_ARG_TYPE_URI

An *A_ARG_TYPE_URI* value is a string formatted as a URI. UI server devices must be able to support values of *A_ARG_TYPE_URI* that are 1024 bytes in length. UI server device support for longer values is optional. All *A_ARG_TYPE_URI* values are never URI escaped and must be UTF-8 encoded.

2.2.4. A_ARG_TYPE_CompatibleUIs

An *A_ARG_TYPE_CompatibleUI*s value is a string formatted as UTF-8 XML representing the list of all UIs that are compatible with a designated Remote UI client device. Remote UI server devices must be able to support values of *A_ARG_TYPE_CompatibleUIs* that are 10k bytes in length. UI server support for longer values is optional.

2.2.5. A_ARG_TYPE_String

A simple string type.

2.2.6. A_ARG_TYPE_Int

A simple integer type.

2.3. Eventing and Moderation

Table 2: Event moderation

Variable Name	Evented	Moderated Event	Max Event Rate [1]	Logical Relation	Min Delta per Event [2]
UIListingUpdate	Yes	Yes	0.5	N/A	N/A
A_ARG_TYPE_DeviceProfile	No	N/A	N/A	N/A	N/A
A_ARG_TYPE_CompatibleUIs	No	N/A	N/A	N/A	N/A
A_ARG_TYPE_String	No	N/A	N/A	N/A	N/A
A_ARG_TYPE_URI	No	N/A	N/A	N/A	N/A
A_ARG_TYPE_Int	No	N/A	N/A	N/A	N/A

[1] Determined by N, where Rate = (Event)/(N seconds).
[2] (N) * (allowedValueRange Step).

2.4. Actions

Immediately following this table is detailed information about these actions, including short descriptions of the actions, the effects of the actions on state variables, and error codes defined by the actions.

Table 3: Actions

Name	Req. or Opt. [1]
GetCompatibleUIs	*R*
SetUILifeTime	*O*

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

[1] R = Required, O = Optional, X = Non-standard.

2.4.1. GetCompatibleUIs

This action retrieves a list of user interfaces that are compatible with a given client device as specified by the client *A_ARG_TYPE_DeviceProfile*. It returns a (possibly empty) list of UIs compatible with the specified device profile.

2.4.1.1. Arguments

Table 4: Arguments for `GetCompatibleUIs()`

Argument	Direction	relatedStateVariable
`InputDeviceProfile`	*IN*	`A_ARG_TYPE_DeviceProfile`
`UIFilter`	*IN*	`A_ARG_TYPE_String`
`UIListing`	*OUT*	`A_ARG_TYPE_CompatibleUIs`

- ***UIFilter*** is a string employed to request additional filtering of the UIListing output. ***UIFilter*** is employed to limit the ***UIListing*** result to UIs that possess specific values of *A_ARG_TYPE_CompatibleUIs* schema elements and attributes. ***UIFilter*** is composed of a comma-separated list of *A_ARG_TYPE_CompatibleUIs* schema elements, attributes and their values (see section 3.1.5 for ***UIFilter*** examples). The ***InputDeviceProfile*** argument always takes precedence over the ***UIFilter*** argument in generating the ***UIListing*** result.

- When the value of ***InputDeviceProfile*** is an empty string, *all* UI listings are returned in the ***UIListing*** output argument.

- When no UIs matching the ***InputDeviceProfile*** are found, **GetCompatibleUIs()** is successful and returns an empty string in the ***UIListing*** output argument..

2.4.1.2. Dependency on State (if any)

2.4.1.3. Effect on State

2.4.1.4. Errors

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
701	Operation Rejected	The Remote UI server has rejected the operation.

702	Invalid Filter Argument	**GetCompatibleUIs()** failed because the ***UIFilter*** argument specifies an invalid filter.

2.4.2. SetUILifetime

Sets the lifetime of UIs for those UIs that permit the default lifetime setting to be modified. A control point may alter the default lifetime setting to change the policy for setting the persistence of the UI after it has been disconnected from a RUI client. After a UI has been disconnected (or was never connected) and the lifetime expires, a UI server will presumably destroy that UI and the state (if any) of the associated application.

For UIs that permit the default lifetime setting to be modified, the lifetime can be set to one of the following values:

- *-1*: the UI has no finite lifetime and can be assumed to persist indefinitely, regardless of whether or not it is connected to a RUI client.
- *0*: the lifetime of the UI has expired.
- *n*: where $n > 0$: the lifetime of the UI will expire *n* minutes from now.

2.4.2.1. Arguments

Table 5: Arguments for SetUILifetime( )

Argument	Direction	relatedStateVariable
UI	*IN*	A_ARG_TYPE_URI
Lifetime	*IN*	A_ARG_TYPE_Int

2.4.2.2. Dependency on State (if any)

None

2.4.2.3. Effect on State (if any)

None

2.4.2.4. Errors

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
701	Operation Rejected	The Remote UI server has rejected the operation.
703	Unsupported Lifetime	**SetUILifetime()** failed because the lifetime specified by the ***Lifetime*** argument is not supported for the specified UI.

2.4.3. Non-Standard Actions Implemented by a UPnP Vendor

To facilitate certification, non-standard actions implemented by UPnP vendors should be included in this service template. The UPnP Device Architecture lists naming requirements for non-standard actions (see the section on Description).

2.4.4. Relationships Between Actions

All actions defined have no specific relationship between them.

2.4.5. Common Error Codes

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

Table 6: Common Error Codes

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
701	Operation Rejected	The Remote UI server has rejected the operation.
702	Invalid Filter	The action failed because a filter specified as an input argument is invalid.
703	Unsupported Lifetime	The action failed because a requested UI lifetime value specified as an input argument is not supported for the specified UI.

3. Theory of Operation

3.1. Example Values of State Variables

3.1.1. A_ARG_TYPE_URI

An *A_ARG_TYPE_URI* value is a string formatted as a URI.

Remote UI server devices and control points that support the optional *A_ARG_TYPE_URI* state variable must be able to support values of *A_ARG_TYPE_URI* that are 1024 bytes in length. Support for longer values is optional.

All *A_ARG_TYPE_URI* values must all be UTF-8 encoded and are never HTTP/URI escaped.

3.1.1.1. UIs

UIs are *A_ARG_TYPE_URI* strings of the following form:

<***PI***>*://*<***SIP***>*[:*<***LPT***>*][/**AID**]*,

where:

PI: A Protocol identifier string. A short string that identifies a peer-to-peer remoting protocol, e.g.: *RDP*, *VNC*, *XRT*, etc.

SIP: An IP address of a server device.
The form of *SIP* is a true IPv4 or IPv6 internet address, not an address *name*.

LPT: A Server port number.

AID: An Application ID. A string that identifies a user interface or a remote-capable application. The value of *AID* can include a session ID.

*UI*s can contain spaces and tabs, but cannot begin with a white space character.

Example:
The following UI corresponds to a Remote UI-enabled DVD browser application available at IP address and port number *1.8.7.2:333.* The XRT2 remoting protocol is used in this case:

XRT2://1.8.7.2:333/DVDui

3.1.2. A_ARG_TYPE_CompatibleUIs

The value of an *A_ARG_TYPE_CompatibleUIs* is an XML block corresponding to a list of applications that are able to generate remote user interfaces, as well as status information on applications associated with UIs.

Remote UI server devices must be able to support values of *A_ARG_TYPE_CompatibleUIs* that are 10k bytes in length. Remote UI server device support for longer values is optional.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<uilist xmlns="urn:schemas-upnp-org:remoteui:uilist-1-0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="urn:schemas-
upnp-org:remoteui:uilist-1-0 CompatibleUIs.xsd">
 <ui>
  <uiID>4560-9876-1265-8758</uiID>
  <name>Music player</name>
  <description>Music browsing and playback application</description>
  <iconList>
   <icon>
    <mimetype>image/png</mimetype>
    <width>40</width>
    <height>40</height>
    <depth>8</depth>
    <url>/icon40.png</url>
   </icon>
   <icon>
    <mimetype>image/png</mimetype>
    <width>160</width>
    <height>160</height>
    <depth>8</depth>
    <url>/icon160.png</url>
   </icon>
   <icon>
    <mimetype>image/jpeg</mimetype>
    <width>40</width>
    <height>40</height>
    <depth>24</depth>
    <url>/icon40.jpeg</url>
   </icon>
   <icon>
    <mimetype>image/jpeg</mimetype>
    <width>160</width>
    <height>160</height>
    <depth>24</depth>
    <url>/icon160.jpeg</url>
   </icon>
  </iconList>
  <fork>true</fork>
  <lifetime>-1</lifetime>
  <protocol shortName="VNC">
   <uri>VNC://1.3.4.5:5910/</uri>
   <uri>VNC://1.5.6.7:5910/</uri>
   <protocolInfo>...opaque...</protocolInfo>
  </protocol>
  <protocol shortName="XRT2">
   <uri>XRT2://1.3.4.5:444/music_player</uri>
   <uri>XRT2://1.5.6.7:444/music_player</uri>
   <protocolInfo>...opaque...</protocolInfo>
  </protocol>
 </ui>
 <ui>
  <uiID>6789-568</uiID>
  <name>DVD Browser</name>
  <protocol shortName="XRT2">
   <uri>XRT2://1.8.7.2:333/DVDui</uri>
```

```
    <protocolInfo>...opaque...</protocolInfo>
   </protocol>
   <protocol shortName="RDP">
    <uri>RDP://1.8.7.2:555</uri>
   </protocol>
  </ui>
</uilist>
```

In the example above, the first remoted application is a music player application and the second is an application for DVD browsing.

Section 4 contains the XSD schema that can be used to validate *A_ARG_TYPE_CompatibleUIs* values.

All values within XML leaf elements are always XML escaped.

<ui>
This required element occurs once for every remote UI advertised by the Remote UI server. Every **<ui>** element contains a detailed description of a single UI.

<uiID>
This required element uniquely identifies a UI.

If two or more servers on the same network serve Remote UIs for the exact same application, the **<uiID>** values must be exactly the same in each Remote UI server *A_ARG_TYPE_CompatibleUIs* listing.

It is recommended that Remote UI server implementations initialize and assign a unique **<uiID>** value for each enumerated Remote UI-enabled application and persist each **<uiID>** value through power cycles, etc. A Remote IU server generating fresh **<uiID>** values after each power cycle event may prevent Remote UI client devices from using the <**uiID**> values to 'bookmark' favorite remote UIs.

A **<uiID>** value must never be longer than 256 bytes in length.

<uiID> values are UTF-8 encoded strings that must not contain: special XML characters “<”, ”>”; quotes; and leading or trailing space or tab characters.

<name>
This required element is a human readable, UTF-8 encoded string no longer than 256 bytes. E.g.: “CompanyX Jukebox”. It is recommended that this friendly name not include the name or IP address of the computer that is serving the user interface.

The value of **<name>** must not contain leading or trailing space or tab characters.

<description>
This optional element contains a human readable, UTF-8 encoded string that is no more that 2048 bytes long. The value of this element defines what the Remote UI-enabled application does, e.g.: “Organize and play your latest and favorite music from a huge selection available from CompanyX ”.

The value of **<description>** must not contain leading or trailing space or tab characters.

<iconList>
This optional element contains a sequence of **<icon>** elements that each describe the form of an icon available to represent this UI. It is recommended to include 40 by 40 and 160 by 160 icons in both the PNG and JPEG image formats.

<icon>
When <**iconList**> is present, it must include at least one <**icon**> child element. The <**icon**> element

includes five optional attributes:
mimetype: MIME type of the icon,
width: (in pixels) of the icon,
height: (in pixels) of the icon,
depth: color depth (in bits) of the icon,
url

<fork>

This optional element indicates whether or not connecting to this UI will fork a new UI to which the connecting client is actually connected. When **<fork>** is *true*, a new UI is spawned by the application in response to a connection request. When **<fork>** is *true*, subsequent UI listings may (depending on the UI server implementation) expose new UIs within the *A_ARG_TYPE_CompatibleUIs* listing corresponding to the UIs that were spawned in response to individual connection requests. When <**fork**> is *false*, new connection requests do not cause a new UI to be spawned and no new UI listings will appear in *A_ARG_TYPE_CompatibleUIs*. The value of **<fork>** must be set to *true* or *false*. The default value of **<fork>** is *false*.

<lifetime>

This optional element indicates the lifetime value set for this UI. A control point may alter the default lifetime setting to change the policy for setting the persistence of the UI after it has been disconnected from a RUI client using the **SetLifetime()** action. After a UI has been disconnected (or was never connected) and the lifetime expires, a UI server will presumably destroy that UI and the state (if any) of the associated application.

For UIs that permit the default lifetime setting to be modified, the lifetime value is set to one of the following values:

- *-1*: the UI has no finite lifetime and can be assumed to persist indefinitely, regardless of whether or not it is connected to a RUI client.
- *0*: the lifetime of the UI has expired.
- *n*: where $n > 0$. The lifetime of the UI will expire *n* minutes from now.

The default value of **<lifetime>** is *–1*.

<protocol>

This required element contains all of the information needed for a specific UI remoting protocol. A **<ui>** element may contain multiple child **<protocol>** elements, indicating support for more than one remoting protocols by a single application.

The **<protocol>** element must define a string value for the required *shortName* attribute. Values for *shortName* corresponding to known Remote UI protocols appear in table 7. Implementations employing one or more of the protocols listed in table 7 must use the *shortName* string values as they are shown in the table (all capital letters).

Table 7: shortName values for known remoting protocols

shortName	Description
HTTP/HTML	The application supports remoting with HTML using HTTP as transport.
RDP	The application supports remoting with Microsoft RDP protocol.

shortName	Description
VNC	The application supports remoting with AT&T VNC protocol.
XRT2	The application supports remoting with Intel XRT2 protocol.
LRDP	The application supports remoting with Nokias' LRDP protocol.
XHT	The application supports remoting with Samsungs' XHT protocol.
SGXML	The application supports remoting with Siemens' Gigaset XML protocol.
UIF	The application supports remoting with Philips UI Fragments protocol.
(Vendor-specific protocol name)	*(Specified by UPnP vendor.)*

Implementations may also use *shortName* to expose support for vendor-specific protocols. Vendor-defined *shortName* values may use any combination of upper or lower case letters. Short name values must be UTF-8 encoded and no longer than 256 bytes.

<uri>

At least one **<uri>** element must be present within each **<protocol>** tag. This element is used to specify a URI that can be used by a Remote UI client to connect to this Remote UI-enabled application. The contents of the **<uri>** element must conform to the rules described in section 3.1.1.1.

Multiple **<uri>** elements are present for the same protocol if the Remote IU server has multiple connectable network interfaces available for that protocol.

All URIs appearing in the *A_ARG_TYPE_CompatibleUIs* listing must be unique.

<protocolInfo>

The optional **<protocolInfo>** tag contains a block of data that is specific to a given remoting protocol. This block of data may contain information that can help in establishing preferences or compatibility between a Remote UI server and a Remote UI client.

Some protocols negotiate all of the session parameters out-of-band upon establishment of the remoting session. **RDP** for example, requires no additional compatibility criteria to be provided by UPnP Remote UI. In these cases the **<protocolInfo>** block is not needed.

3.1.3. A_ARG_TYPE_DeviceProfile

A_ARG_TYPE_DeviceProfile values are XML-formatted strings used by the Remote UI client device to represent its list of all remoting protocols supported by the Remote UI client device. The format of the device profile is UTF-8 encoded XML.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<deviceprofile xmlns="urn:schemas-upnp-org:remoteui:devprofile-1-0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="urn:schemas-
upnp-org:remoteui:devprofile-1-0 DeviceProfile.xsd">
  <maxHoldUI>5</maxHoldUI>
  <protocol shortName="LRDP">
    <protocolInfo>LRDP:image 1500 UDP beep:sendonly</protocolInfo>
  </protocol>
  <protocol shortName="XHT">
    <protocolInfo> (opaque) </protocolInfo>
  </protocol>
  <protocol shortName="RDP"/>
  <protocol shortName="XRT2">
<protocolInfo>version=2.1,displayWidth=640,displayHeight=480,imageEncoding=JPEG&PNG,se
rverVolumeControl=TRUE,videoViewPortRequired=TRUE</protocolInfo>
  </protocol>
</deviceprofile>
```

A_ARG_TYPE_DeviceProfile values are validated with the XSD schema in section 5.

<MaxHoldUI>

This optional element defines how many user interfaces can be put on hold or in the background. If the Remote UI client does not support placing user interfaces on hold, the **<MaxHoldUI>** value is set to *0*.

The default value of **<MaxHoldUI>** is *0*.

<protocol>

This required tag contains all the information needed for a specific UI remoting protocol. Multiple child **<protocol>** elements may be used to indicate support for more than one remoting protocol.

The **<protocol>** tag must define a string value for the required *shortName* attribute. Values for *shortName* corresponding to known Remote UI protocols appear in table 7. Implementations employing one or more of the protocols listed in table 7 must use the *shortName* string values as they are shown in the table (all capital letters).

Implementations may also use *shortName* to expose support for vendor-specific protocols. Vendor-defined *shortName* values may use any combination of upper or lower case letters. Short name values must be UTF-8 encoded and no longer than 256 bytes.

<protocolInfo>

The optional **<protocolInfo>** tag contains a block of data that is specific to a given remoting protocol. This block of data may contain information that can help in establishing preferences or compatibility between a Remote UI server and a Remote UI client.

Some protocols negotiate all of the session parameters out-of-band upon establishment of the remoting session. **RDP** for example, requires no additional compatibility criteria to be provided by UPnP Remote UI. In these cases the **<protocolInfo>** block is not needed.

3.1.4. UIListingUpdate

The optional *UILisitingUpdate* state variable is composed of a list of unique identifiers corresponding to user interfaces that have changed since the last event. Each user interface is demarcated within the list by <uiID> tags. Each line in the listing, including the last line, must be terminated with a carriage-return character and a linefeed character. *UILisitingUpdate* values may consist of an empty string when the event is issued by the Remote UI server for the first time.

Example:

```
<uiID>4560-9876-1265-8758</uiID>
<uiID>6294-2053-A34E-A432</uiID>
<uiID>6789-568</uiID>
```

3.1.5. UIFilter Examples

The **GetCompatibleUIs**() action returns UI listings that are a match for the ***InputDeviceProfile*** argument. Additional filtering of the UIs returned in the ***UIListing*** output argument can be accomplished through the use of the ***UIFilter*** input argument.

UIFilter is composed of a comma-separated list of *A_ARG_TYPE_CompatibleUIs* schema elements, attributes and their values.

Example 1:
The following ***UIFilter*** argument causes only UI listings with 'music' in the value of the required <**name**> element to be returned in the ***UIListings*** output argument. It also causes the optional <**description**> element to be included, regardless of it's value:

"name="*music*",description="*"",

where the wild-card '*' character generates a match on an unlimited number of UTF-8 characters.

Example 2:
The following ***UIFilter*** argument causes only UI listings with 'png' somewhere in the value of the optional <**icon**> element attribute **mimetype** to be returned. Note that a requested **mimetype** attribute is only returned if there is an accompanying <**icon**> element actually present in the UI listings that match ***InputDeviceProfile***. The ***UIFilter*** argument is ignored otherwise:

"icon@mimetype="*png*""

If the entire ***UIFilter*** parameter is equal to "*", all optional elements and attributes (when present) and their values, are returned in ***UIListing***.

If a value for an optional *A_ARG_TYPE_CompatibleUIs* schema element or attribute is specified in ***UIFilter*** that is not actually present in any of the UI listings that are a match for the ***InputDeviceProfile*** argument, then the optional element or attribute is ignored. **GetCompatibleUIs**() returns successfully in these cases.

The ***InputDeviceProfile*** argument always takes precedence over the ***UIFilter*** argument in generating the ***UIListing*** result.

Elements and attributes required by the *A_ARG_TYPE_CompatibleUIs* schema to be present in the ***UIListing*** output argument are always returned. UI server implementations do not return optional *A_ARG_TYPE_CompatibleUIs* schema elements and attributes unless explicitly requested in the ***UIFilter*** input argument.

4. A_ARG_TYPE_CompatibleUIs XSD Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="urn:schemas-upnp-org:remoteui:uilist-1-0"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified" id="uilist">
 <xs:element name="uilist">
  <xs:complexType>
   <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:element name="ui">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="uiID" type="xs:string"/>
       <xs:element name="name" type="xs:string"/>
       <xs:element name="description" type="xs:string" minOccurs="0"/>
       <xs:element name="iconList" minOccurs="0">
        <xs:complexType>
         <xs:sequence maxOccurs="unbounded">
          <xs:element name="icon">
           <xs:complexType>
            <xs:sequence>
             <xs:element name="mimetype" type="xs:string"/>
             <xs:element name="width" type="xs:positiveInteger"/>
             <xs:element name="height" type="xs:positiveInteger"/>
             <xs:element name="depth" type="xs:positiveInteger"/>
             <xs:element name="url" type="xs:anyURI"/>
            </xs:sequence>
           </xs:complexType>
          </xs:element>
         </xs:sequence>
        </xs:complexType>
       </xs:element>
       <xs:element name="fork" type="xs:boolean" default="false" minOccurs="0"/>
       <xs:element name="lifetime" type="xs:integer" default="-1" minOccurs="0"/>
       <xs:element name="protocol" maxOccurs="unbounded">
        <xs:complexType>
         <xs:sequence>
          <xs:element name="uri" type="xs:anyURI" nillable="false"
maxOccurs="unbounded"/>
          <xs:element name="protocolInfo" type="xs:anyType" nillable="true" minOccurs="0"/>
         </xs:sequence>
         <xs:attribute name="shortName" type="xs:string" use="required" form="unqualified"/>
        </xs:complexType>
       </xs:element>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

5. DeviceProfile XSD Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="urn:schemas-upnp-org:remoteui:devprofile-1-0"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified" id="deviceprofile">
  <xs:element name="deviceprofile">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="maxHoldUI" type="xs:unsignedInt" default="0" minOccurs="0"/>
        <xs:element name="protocol" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="protocolInfo" type="xs:anyType" nillable="true" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="shortName" type="xs:string" use="required" form="unqualified"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

6. XML Service Description

```
<?xml version="1.0" encoding="UTF-8"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
 <specVersion>
  <major>1</major>
  <minor>0</minor>
 </specVersion>
 <actionList>
  <action>
   <name>GetCompatibleUIs</name>
   <argumentList>
    <argument>
     <name>InputDeviceProfile</name>
     <direction>in</direction>
     <relatedStateVariable>A_ARG_TYPE_DeviceProfile</relatedStateVariable>
    </argument>
    <argument>
     <name>UIFilter</name>
     <direction>in</direction>
     <relatedStateVariable>A_ARG_TYPE_String</relatedStateVariable>
    </argument>
    <argument>
     <name>UIListing</name>
     <direction>out</direction>
     <relatedStateVariable>A_ARG_TYPE_CompatibleUIs</relatedStateVariable>
    </argument>
   </argumentList>
  </action>
  <action>
   <name>SetUILifetime</name>
   <argumentList>
    <argument>
     <name>UI</name>
     <direction>in</direction>
     <relatedStateVariable>A_ARG_TYPE_URI</relatedStateVariable>
    </argument>
    <argument>
     <name>Lifetime</name>
     <direction>in</direction>
     <relatedStateVariable>A_ARG_TYPE_Int</relatedStateVariable>
    </argument>
   </argumentList>
  </action>
 </actionList>
 <serviceStateTable>
  <stateVariable sendEvents="yes">
   <name>UIListingUpdate</name>
   <dataType>string</dataType>
  </stateVariable>
  <stateVariable sendEvents="no">
   <name>A_ARG_TYPE_DeviceProfile</name>
   <dataType>string</dataType>
  </stateVariable>
```

```
    <stateVariable sendEvents="no">
      <name>A_ARG_TYPE_URI</name>
      <dataType>uri</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>A_ARG_TYPE_CompatibleUIs</name>
      <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>A_ARG_TYPE_String</name>
      <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>A_ARG_TYPE_Int</name>
      <dataType>int</dataType>
    </stateVariable>
  </serviceStateTable>
</scpd>
```