Car Connectivity Consortium

MirrorLink®

UPnP Application Server Service

Version 1.1.6
(CCC-TS-024)

Version History

Version	Date	Comment
1.0	06 October 2010	Public release
1.0.1	26 June 2011	Approved Version
1.1	31 March 2012	Approved Version
1.1.1	16 October 2012	Approved Errata Version
1.1.2	05 December 2012	Approved Errata Version
1.1.3	05 March 2013	Approved Errata Version
1.1.4	18 June 2013	Approved Errata Version
1.1.5	29 August 2013	Approved Errata Version
1.1.6	05 November 2013	Approved Errata Version

Contributors

Benesch, Matthias Daimler AG
Bose, Raja, Nokia Corporation
Brakensiek, Jörg (Editor) Nokia Corporation

Trademarks

Legal Notice

The copyright in this Specification is owned by the Car Connectivity Consortium LLC ("CCC LLC"). Use of this Specification and any related intellectual property (collectively, the "Specification"), is governed by these license terms and the CCC LLC Limited Liability Company Agreement (the "Agreement").

Use of the Specification by anyone who is not a member of CCC LLC (each such person or party, a "Member") is prohibited. The legal rights and obligations of each Member are governed by the Agreement and their applicable Membership Agreement, including without limitation those contained in Article 10 of the LLC Agreement.

CCC LLC hereby grants each Member a right to use and to make verbatim copies of the Specification for the purposes of implementing the technologies specified in the Specification to their products ("Implementing Products") under the terms of the Agreement (the "Purpose"). Members are not permitted to make available or distribute this Specification or any copies thereof to non-Members other than to their Affiliates (as defined in the Agreement) and subcontractors but only to the extent that such Affiliates and subcontractors have a need to know for carrying out the Purpose and provided that such Affiliates and subcontractors accept confidentiality obligations similar to those contained in the Agreement. Each Member shall be responsible for the observance and proper performance by such of its Affiliates and subcontractors of the terms and conditions of this Legal Notice and the Agreement. No other license, express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

Any use of the Specification not in compliance with the terms of this Legal Notice, the Agreement and Membership Agreement is prohibited and any such prohibited use may result in termination of the applicable Membership Agreement and other liability permitted by the applicable Agreement or by applicable law to CCC LLC or any of its members for patent, copyright and/or trademark infringement.

THE SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS, AND COMPLIANCE WITH APPLICABLE LAWS.

Each Member hereby acknowledges that its Implementing Products may be subject to various regulatory controls under the laws and regulations of various jurisdictions worldwide. Such laws and regulatory controls may govern, among other things, the combination, operation, use, implementation and distribution of Implementing Products. Examples of such laws and regulatory controls include, but are not limited to, road safety regulations, telecommunications regulations, technology transfer controls and health and safety regulations. Each Member is solely responsible for the compliance by their Implementing Products with any such laws and regulations and for obtaining any and all required authorizations, permits, or licenses for their Implementing Products related to such regulations within the applicable jurisdictions.

Each Member acknowledges that nothing in the Specification provides any information or assistance in connection with securing such compliance, authorizations or licenses**.**

NOTHING IN THE SPECIFICATION CREATES ANY WARRANTIES, EITHER EXPRESS OR IMPLIED, REGARDING SUCH LAWS OR REGULATIONS. ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY INTELLECTUAL PROPERTYRIGHTS OR FOR NONCOMPLIANCE WITH LAWS, RELATING TO USE OF THE SPECIFICATION IS EXPRESSLY DISCLAIMED. BY USE OF THE SPECIFICATION, EACH MEMBER EXPRESSLY WAIVES ANY CLAIM AGAINST CCC LLC AND ITS MEMBERS RELATED TO USE OF THE SPECIFICATION.

CCC LLC reserve the right to adopt any changes or alterations to the Specification as it deems necessary or appropriate.

Contents

Version History **2**
Contributors **2**
Legal Notice **3**
Contents **4**
List of Tables **6**
1 Overview and Scope **7**
1.1 Introduction 7
2 Service Modeling Definitions **8**
2.1 Service Type 8
2.2 State Variables 8
2.2.1 AppStatusUpdate 8
2.2.2 AppListUpdate 8
2.2.3 A_ARG_TYPE_AppStatus 9
2.2.4 A_ARG_TYPE_AppID 9
2.2.5 A_ARG_TYPE_ProfileID 9
2.2.6 A_ARG_TYPE_AppList 9
2.2.7 A_ARG_TYPE_URI 14
2.2.8 A_ARG_TYPE_String 15
2.2.9 A_ARG_TYPE_Bool 15
2.2.10 A_ARG_TYPE_INT 15
2.2.11 A_ARG_TYPE_AppCertificateInfo 15
2.3 Eventing and Moderation 18
2.4 Supporting Multiple Client Profiles 18
2.5 Actions 18
2.5.1 GetApplicationList 19
2.5.2 LaunchApplication 20
2.5.3 TerminateApplication 21
2.5.4 GetApplicationStatus 23
2.5.5 GetApplicationCertificateInfo 24
2.5.6 GetCertifiedApplicationsList 25
2.5.7 GetAppCertificationStatus 25
2.5.8 Relationships Between Actions 26
2.5.9 Error Code Summary 27
3 Theory of Operation **29**
3.1 Use of Quotation Marks 29
3.2 Identification of Applications from the A_ARG_TYPE_AppList 29
3.2.1 Identifying the VNC Server 29
3.2.2 Identifying Remote VNC based Applications 29
3.2.3 Identifying Audio Links 29
3.2.4 Identifying Common Data Bus 30
3.2.5 Identifying Device Attestation Protocol Server 31
3.3 Example Values of AppListingFilter 31
3.4 Example Values of AppCertFilter 32

3.5 Example Values of State Variables 33
3.5.1 AppStatusUpdate 33
3.5.2 AppListUpdate 33
3.5.3 A_ARG_TYPE_AppStatus 33
3.5.4 A_ARG_TYPE_AppList 34
3.6 XML Signature Minimum Set 36
3.7 Handling of Applications Available via Home Screen Application 37
4 XSD Schema 38
4.1 A_ARG_TYPE_AppStatus XSD Schema 38
4.2 A_ARG_TYPE_AppList XSD Schema 40
4.3 A_ARG_TYPE_AppCertificateInfo XSD Schema 44
5 XML Service Description 46
6 Appendix A – Application Context Information 51
Trust Level 51
Application Categories 51
Content Categories 53
Content Rules 53
7 References 55

List of Tables

Table 2-1: Service State Variables 8
Table 2-2: Structure of A_ARG_TYPE_AppStatus 9
Table 2-3: Structure of A_ARG_TYPE_AppList 10
Table 2-4: Supported Remote Access Protocols 13
Table 2-5: Remote Access Protocol Format 14
Table 2-6: Used Elements in A_ARG_TYPE_AppList 14
Table 2-7: URI Field values for Supported Remote Access Protocols 15
Table 2-8: Structure of A_ARG_TYPE_AppCertificateInfo 15
Table 2-9: Eventing and Moderation 18
Table 2-10: TmApplicationServer Service Actions 19
Table 2-11: Arguments for GetApplicationList 19
Table 2-12: Error Codes for GetApplicationList 20
Table 2-13: Arguments for LaunchApplication 20
Table 2-14: Error Codes for LaunchApplication 21
Table 2-15: Arguments for TerminateApplication 22
Table 2-16: Error Codes for TerminateApplication 23
Table 2-17: Arguments for GetApplicationStatus 23
Table 2-18: Error Codes for GetApplicationStatus 24
Table 2-19: Arguments for GetApplicationCertificateInfo 24
Table 2-20: Error Codes for GetApplicationCertificateInfo 24
Table 2-21: Arguments for GetCertifiedApplicationsList 25
Table 2-22: Error Codes for GetCertifiedApplicationsList 25
Table 2-23: Arguments for GetAppCertificationStatus 26
Table 2-24: Error Codes for GetAppCertificationStatus 26
Table 2-25: Error Code Summary 27
Table 6-1: Trust Level 51
Table 6-2: Application Categories 51
Table 6-3: Content Categories for Visual Content 53
Table 6-4: Content Categories for Audio Content 53
Table 6-5: Content Rules 53

1 Overview and Scope

This service definition is compliant with the UPnP Device Architecture version 1.0 [1]. It defines a service type referred to herein as TmApplicationServer service.

The specification lists a series of requirements, either explicitly or within the text, which are mandatory elements for compliant solutions. Recommendations are given to ensure optimal usage and to provide suitable performance. All recommendations are OPTIONAL.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are following the notation as described in RFC 2119 [1].

- MUST: This word, or the terms "REQUIRED" or "SHALL", means that the definition is an absolute requirement of the specification.
- MUST NOT: This phrase, or the phrase "SHALL NOT", means that the definition is an absolute prohibition of the specification.
- SHOULD: This word, or the adjective "RECOMMENDED", means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- SHOULD NOT: This phrase, or the phrase "NOT RECOMMENDED" means that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- MAY: This word, or the adjective "OPTIONAL", means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

1.1 Introduction

The TmApplicationServer service is a UPnP service that allows UPnP Control Points to remotely launch and terminate applications on MirrorLink Server devices. Through this service, UPnP control points can provide more fine-grained control and access to specific remote applications.

2 Service Modeling Definitions

2.1 Service Type

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

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

TmApplicationServer service is used herein to refer to this service type.

2.2 State Variables

Table 2-1: Service State Variables

Variable Name	Req. or Opt.	Data Type	Allowed Value	Default Value	Eng. Units
AppStatusUpdate	R	string	Undefined	Empty string	N/A
AppListUpdate	R	string	Undefined	Empty string	N/A
A_ARG_TYPE_AppStatus	R	string	Undefined	Empty string	N/A
A_ARG_TYPE_AppID	R	string	Undefined	Empty string	N/A
A_ARG_TYPE_ProfileID	R	ui4	Undefined	0	N/A
A_ARG_TYPE_URI	R	string	Undefined	Empty string	N/A
A_ARG_TYPE_AppList	R	string	Undefined	Empty string	N/A
A_ARG_TYPE_String	R	string	Undefined	Empty string	N/A
A_ARG_TYPE_Bool	R	string	true | false	false	N/A
A_ARG_TYPE_INT	R	ui4	Undefined	0	N/A
A_ARG_TYPE_AppCertificateInfo	R	string	Undefined	Empty string	N/A

R = *REQUIRED*, O = *OPTIONAL*, X = *Non-standard*

2.2.1 AppStatusUpdate

A string formatted as UTF-8 represents the list of application identifiers (appIDs) of applications whose status has changed. The string consists of a comma-separated list of appIDs identifying applications whose status has changed. Each entry in the list is of the type A_ARG_TYPE_AppID.

This state variable is evented, implying that clients can subscribe to receive notifications every time the variable changes using UPnP standardized eventing mechanisms. It is important to note that this variable only contains the appIDs of those applications, whose status has changed since the last time an event notification was sent out.

On receiving an AppStatusUpdate event, the MirrorLink UPnP Control Point can query the application status of specific applications in the list by invoking the GetApplicationStatus action.

AppStatusUpdate value will consist of a comma separated list of all application identifiers (appIDs) of applications listed in A_ARG_TYPE_AppList when the event is issued by the TmApplicationServer service for the first time.

2.2.2 AppListUpdate

A string formatted as UTF-8 represents a list of application identifiers (appIDs) of applications whose entries have changed in the application listing. The string consists of a comma-separated list of appIDs identifying applications whose status has changed. Each entry in the list is of the type A_ARG_TYPE_AppID.

This state variable is OPTIONAL. It is evented, implying that clients can subscribe to receive notifications every time the variable changes using UPnP standardized eventing mechanisms. It is important to note that this variable only contains the appIDs of those applications, whose entries in the application list have changed since the last time an event notification was sent out.

On receiving an AppListUpdate event, a MirrorLink UPnP Control Point can retrieve the application list by invoking the GetApplicationList action and specifying the appropriate filter using the appListingFilter input argument.

AppListUpdate value will consist of a comma separated list of all application identifiers (appIDs) of applications listed in A_ARG_TYPE_AppList when the event is issued by the TmApplicationServer service for the first time.

2.2.3 A_ARG_TYPE_AppStatus

A string formatted as UTF-8 XML represents the status of a specific application or alternatively providing the status of all applications, which can be controlled remotely. Its structure is given in the following table.

Table 2-2: Structure of A_ARG_TYPE_AppStatus

Element	Description	Parent	Availability
appStatusList	Indicates list of application status updates	-	Required
appStatus*	Indicates status record corresponding to an application	appStatusList	Required
appID	Unique ID of the application (A_ARG_TYPE_AppID)	appStatus	Required
status*	Entry corresponding to an instance of the application running under a specific client profile	appStatus	Required
profileID	Profile Identifier of the client profile (A_ARG_TYPE_ProfileID)	status	Required
statusType	String representing status of application: {Foreground | Background | Notrunning} (A_ARG_TYPE_String)	status	Required

The elements marked with a (*) can have multiple instances.

2.2.4 A_ARG_TYPE_AppID

A UTF-8 encoded string represents an unsigned 32-bit integer in hexadecimal format (with '0x' prefix) which denotes the unique application identifier.

The MirrorLink Server MUST use the unsigned integer value of a variable of this type within any action. I.e. comparing the values of two A_ARG_TYPE_AppID variables MUST be done based on the unsigned integer value and not based on a specific character representation.

Therefore, the following two A_ARG_TYPE_AppID values are identical:

- `0x45ab` and `0x45AB` (case insensitivity of the hexadecimal numbers)
- `0x45ab` and `0X45ab` (case insensitivity of the 0x)
- `0x00001234` and `0x001234` (leading zeros do not matter)

Note: The application identifier SHOULD be the same over time for the same application (e.g. should survive a reboot or MirrorLink reconnect), to allow the MirrorLink Client to implement a Last-Mode behavior.

2.2.5 A_ARG_TYPE_ProfileID

An unsigned 32-bit integer greater than or equal to 0, represents a unique profile identifier. Its value is set equal to 0 by default.

2.2.6 A_ARG_TYPE_AppList

A string formatted as UTF-8 XML represents the list of all applications that are available for remote control and access through the TmApplicationServer service. Its structure is given in the following table. Server devices MUST be able to support values of *A_ARG_TYPE_AppList* up to 10 KiloBytes in length.

Table 2-3: Structure of A_ARG_TYPE_AppList

Element	Description	Parent	Availability
appList	List of all available remote applications	-	Required
app*	Entry describing one remote application	appList	Optional
appID	Unique application ID. MUST be non-zero. (A_ARG_TYPE_AppID)	app	Required
name	Application name (A_ARG_TYPE_String)	app	Required
variant	Comma separated list of application variant names. Variant names can be a string which identifies a specific version of the application or identifies a specific head unit platform targeted by the application. Note: Variant names are defined by automotive manufacturers and OEMs for their respective product brands. (A_ARG_TYPE_String)	app	Optional
provider Name	Name of the application provider (A_ARG_TYPE_String)	app	Optional
provider URL	URL of the application provider's website (A_ARG_TYPE_URI)	app	Optional
description	Text description of application (A_ARG_TYPE_String)	app	Optional
iconList	List of available application icons	app	Optional
icon*	Describes an application icon The MirrorLink server MUST include an icon for all applications with <protocolID> = VNC.	iconList	Optional
mimetype	Type of icon image (see below). • At least one icon type SHOULD support a transparent background, such as mimetype `/image/png`. • One icon type MUST be either mimetype `image/png` and color depth 24 or a mimetype and color depth identical to values set in the client icon preferences as specified using the TmClientProfileServer:1 service's SetClientProfile action. • The MirrorLink client MUST have support for displaying icons with mimetype `image/png` and color depth 24. (A_ARG_TYPE_String)	icon	Required
width	Width of icon (A_ARG_TYPE_INT)	icon	Required
height	Height of icon (A_ARG_TYPE_INT)	icon	Required
depth	Color depth of icon (A_ARG_TYPE_INT)	icon	Required
url	URL where icon is available MirrorLink Client MUST use HTTP-GET to access the icon behind the URL. (A_ARG_TYPE_URI)	icon	Required
allowed ProfileIDs	Comma separated list of client profile identifier (A_ARG_TYPE_ProfileID), for which this application can currently be launched and executed; If this list is not explicitly specified then it MUST be assumed that the application can be executed using any profile with a valid profileID.	app	Optional

Element	Description	Parent	Availability
	In case the application cannot be launched at a given time via any of the available profiles, then the value of this element MUST be set equal to -1 (A_ARG_TYPE_String) **Default**: "0"		
remoting Info	Information about the remoting protocol used to interact with the application after it is launched	app	Required
protocolID	Protocol Identifier of the remoting protocol that will be used to access the application (see Table 2-4 for list of supported protocols) (A_ARG_TYPE_String)	remoting Info	Required
format	Format of data being transferred using the remoting protocol (see Table 2-5 for details) (A_ARG_TYPE_String)	remoting Info	Optional
direction	Direction of the content stream. A_ARG_TYPE_String with one of the following values: • "out" – Content streaming from the MirrorLink server device to the client • "in" – Content streaming from the MirrorLink client to the server. • "bi" – Content streaming in both directions between the MirrorLink server and client. **Default**: "out"	remoting Info	Optional
audioIPL	Audio Initial Playback Latency (A_ARG_TYPE_INT) **Default**: "4800"	remoting Info	Optional
audioMPL	Audio Maximum Playback Length (A_ARG_TYPE_INT) **Default**: "9600"	remoting Info	Optional
app Certificate URL	URL where application certificate+ is available. Head Unit uses the certificate to verify trust worthiness of the application information provided. MirrorLink Client MUST use HTTP-GET to access the certificate behind the URL. (A_ARG_TYPE_URI)	app	Optional
appInfo	Information about the listed application	app	Optional
app Category	Application category (A UTF-8 encoded string representing an unsigned 32-bit integer in hexadecimal format (with '0x' prefix).) Values are defined in Appendix A. **Default**: "0x00000000"	appInfo	Optional
trustLevel	Trust level of the contents of the appInfo element (A UTF-8 encoded string representing an unsigned 16-bit integer in hexadecimal format (with '0x' prefix).) Values are defined in Appendix A **Default**: "0x0000"	appInfo	Optional
displayInfo	Information about display content of the listed application, in case it provides a displayable user interface	app	Optional
content Category	Visual content categories used (A UTF-8 encoded string representing an unsigned 32-bit integer in hexadecimal format (with '0x' prefix).) Values are defined in Appendix A. **Default**: "0x00000000"	display Info	Optional

Element	Description	Parent	Availability
content Rules	Visual content rules followed (A UTF-8 encoded string representing an unsigned 32-bit integer in hexadecimal format (with '0x' prefix).) Values are defined in Appendix A. Deprecated use for applications coming with an application certificate issued from CCC. **Default**: "0x00000000"	display Info	Optional
orientation	Display orientations supported. A_ARG_TYPE_String with one of the following values: • "landscape" – Support for landscape only • "portrait" – Support for portrait only • "both" – Support for both, landscape and portrait • "mixed" – Support for landscape only and portrait only is mixed **Default**: "both"	display Info	Optional
trustLevel	Trust level of the displayInfo element (A UTF-8 encoded string representing an unsigned 16-bit integer in hexadecimal format (with '0x' prefix).) Values are defined in Appendix A. **Default**: "0x0000"	display Info	Optional
audioInfo	Information about audio content of the listed application, in case it provides an audio interface	app	Optional(++)
audioType	Audio type A_ARG_TYPE_String with one of the following values: • "phone" – Phone call audio • "application" – Generic application audio • "all" – Phone and application audio • "none" – no audio	audioInfo	Required
content Category	Audio content categories used (A UTF-8 encoded string representing an unsigned 32-bit integer in hexadecimal format (with '0x' prefix).) Values are defined in Appendix A.	audioInfo	Required
content Rules	Audio content rules followed (A UTF-8 encoded string representing an unsigned 32-bit integer in hexadecimal format (with '0x' prefix).) Values are defined in Appendix A. Deprecated use for applications coming with an application certificate issued from CCC. **Default**: "0x00000000"	audioInfo	Optional
trustLevel	Trust level of the audioInfo element (A UTF-8 encoded string representing an unsigned 16-bit integer in hexadecimal format (with '0x' prefix).) Values are defined in Appendix A. **Default**: "0x0000"	audioInfo	Optional
resource Status	Application resource status In case the remote application is using a resource, which is subject to access control (e.g. an audio source or sink), this element will define the current status having one of the following values (A_ARG_TYPE_String): • "free" – Resource is free. Can be used by the MirrorLink client • "busy" – Resource already used. Resource assignment can be overridden by a client's invocation LaunchApplication action.	app	Optional

Element	Description	Parent	Availability
	• "NA" – Resource already used. Resource assignment cannot be overridden by a LaunchApplication action invoked by a client. **Default**: "free"		
Signature	XML signature over entire contents of the appList element. This is done as specified in [4]. The key used in calculating the signature MUST be the private part of the application-specific key which public part was bound to the attestation of UPnP-Server component. (The public part can be used to verify the signature.) The Reference element of the XML signature MUST point to appList element. The SignatureMethod MUST be RSA with SHA1. The KeyInfo element MAY be omitted. The mechanism for generation, exchange and maintainance of keys is out of scope for this specification.	appList	Optional

The elements marked with a (*) can have multiple instances.
(++) The *audioInfo* element MUST be included into the advertisement on any RTP Client, RTP Server, BT HFP or BT A2DP module.

The MirrorLink specification uses UTF-8 encoded strings, representing an unsigned 16- or 32-bit integer in hexadecimal format (with '0x' prefix). The MirrorLink Server and Client MUST use the unsigned integer value of a variable of this type within any action or response. I.e. comparing the values of two such variables MUST be done based on the unsigned integer value and not based on a specific character representation.

Therefore, the following two values are identical:

- `0x45ab` and `0x45AB` (case insensitivity of the hexadecimal numbers)
- `0x45ab` and `0X45ab` (case insensitivity of the 0x)
- `0x00001234` and `0x001234` (leading zeros do not matter)

The `protocolID` element in A_ARG_TYPE_AppList is a string formatted as UTF-8 XML represents the remote access protocol of a specific application, which can be controlled remotely. The following table specifies the supported remote access prototocols, supported from the TmApplicationServer:1 service.

Table 2-4: Supported Remote Access Protocols

protocolID	Protocol Name and Description
VNC	Virtual Networking Computing
RTP	Real Time Protocol
BTA2DP	Bluetooth Advanced Audio Distribution Profile
BTHFP	Bluetooth Hands Free Profile
DAP	Device Attestation Protocol
CDB	Common Data Bus
NONE	Used to indicate that application does not have any additional out-of-band connection using a remote access protocol
<VendorName-ProtocolName>	Vendor Specific Protocol Name. Note that the vendor name MUST be appended in front of the protocol name and separated by an '-' (Hyphen)

The `format` element in A_ARG_TYPE_AppList is a string formatted as UTF-8 XML represents additional format information for dedicated remote access protocols. The following table specifies the Remote Access Protocol format information.

Table 2-5: Remote Access Protocol Format

protocolID	Remote Access Protocol Format description
VNC	Not used
RTP	Comma separated list of supported RTP payload types. **Default**: "99"
BTA2DP	Not used
BTHFP	Not used
DAP	Version number of the DAP protocol **Allowed Values**: "1.0" or "1.1" **Default**: "1.1"
CDB	Version number of the CDB protocol **Allowed Values**: "1.1" **Default**: "1.1"
NONE	Not used
<VendorName-ProtocolName>	Vendor Specific

The A_ARG_TYPE_AppList contains many OPTIONAL elements. Elements, which are used for specific remote access protocols, are given in the following table.

Table 2-6: Used Elements in A_ARG_TYPE_AppList

Remote Access Protocol	VNC	DAP	CDB	BTHFP BTA2DP	RTP
remotingInfo	Used	Used	Used	Used	Used
appInfo	Used	Used	Used	-	Used
displayInfo	Used	-	-	-	-
audioInfo	Used	-	-	Used	Used
resourceStatus	Used	-	-	Used	-

An application with a `protocolID` of "VNC" MAY use the `resourceStatus` value "NA" to indicate that the particular application is accessible to launch, via a separately advertised Home Screen application.

2.2.7 A_ARG_TYPE_URI

A string encoded as UTF-8 represents a URI according to the following format, given in [3]:

```
      foo://example.com:8042/over/there?name=ferret#nose
      \_/   \______________/\_________/ \_________/ \__/
       |           |            |            |        |
    scheme     authority       path        query   fragment
```

with the authority being defined as

```
      example.com:8042
      \_________/ \__/
           |        |
        host     port
```

The values of `scheme`, `host` and `port` fields will differ based on the specific remoting protocol being used, as given in the following table. The `port` field is OPTIONAL for BTHFP and BTA2DP protocol identifiers. The `port` field MUST be present for other non-vendor specific remote protocol identifiers. A_ARG_TYPE_URI is not specified for any vendor specific remote protocol identifier.

Table 2-7: URI Field values for Supported Remote Access Protocols

scheme	host	port	path, query, fragment
VNC	IP address of the VNC server (MANDATORY)	Port number of the VNC server (MANDATORY)	Not used
DAP	IP address of the DAP server (MANDATORY)	Port number of the DAP server (MANDATORY)	Not used
RTP	IP address of the RTP server or client (MANDATORY)	Port number of the RTP server or client (MANDATORY)	Not used
BTA2DP	ASCII string of the 48-bit Bluetooth address in hexadecimal notation (MANDATORY)	Stream End Point Identifier (SEID) in hexadecimal notation (OPTIONAL)	Not used
BTHFP	ASCII string of the 48-bit Bluetooth address in hexadecimal notation (MANDATORY)	RFCOMM channel in hexadecimal notation (OPTIONAL)	Not used
CDB	IP address of the CDB endpoint (MANDATORY)	Port number of the CDB endpoint (MANDATORY)	Not used
http	IP address of the resource being accessed	Port number of the resource being accessed	Used MUST be available, in case the URI is used via HTTP-GET to retrieve the resource.

The MirrorLink client MUST use the **`http`** `schema` to the `host` and `port` of the UPnP Application Server Service's URL, if all `schema`, `host` and `port` entries are missing from the URI.

2.2.8 A_ARG_TYPE_String

A simple string type (UTF-8).

2.2.9 A_ARG_TYPE_Bool

A simple Boolean string which can either have the value 'true' or 'false'.

2.2.10 A_ARG_TYPE_INT

A simple unsigned 32-bit integer represented in decimal (base 10) format.

2.2.11 A_ARG_TYPE_AppCertificateInfo

A string formatted as UTF-8 XML representing an application certificate. The format is given in the following table. Its structure is given in the following table.

Table 2-8: Structure of A_ARG_TYPE_AppCertificateInfo

Element	Description	Parent	Availability
certification	Application certification information	-	Optional
appID	Application identifier (appID) of the requested application. MirrorLink Client MUST check, whether the appID is equal to the one requested.	certification	Required

Element	Description	Parent	Availability
nonce	Random DAP nonce, provided from the MirrorLink Client during the last DAP request. 20-byte random number Base64-encoded. Empty string indicates, that MirrorLink Client has not used DAP during the active MirrorLink session. (A_ARG_TYPE_String)	certification	Required
appUUID	UUID of the application Unique application identifier (A_ARG_TYPE_String)	certification	Optional
entity*	Certifying entity (e.g. CCC, car OEM, HU OEM) The application MUST be considered not certified, if this field is not available.	certification	Optional
name	Entity name CCC certified applications MUST have "CCC" as an entity name. (A_ARG_TYPE_String)	entity	Required
targetList	Target **Default**: All target are certified	entity	Optional
target*	Target name Comma separated list of MirrorLink Client vendor specific values. Might be interpreted as a White and/or a Black list. (A_ARG_TYPE_String)	targetList	Required
restricted+	Comma separated list of locales, where restricted use is certified (A_ARG_TYPE_String)	entity	Required
nonRestricted+	Comma separated list of locales, where non-restricted use is certified (A_ARG_TYPE_String)	entity	Required
serviceList	List of allowed data services **Default**: All services are certified	entity	Optional
service+*	Service name (A_ARG_TYPE_String)	serviceList	Required
properties	Application properties Contains an UTF-8 XML representation of certified application properties. The XML representation is out-of-scope of this specification. **Default**: Empty string (A_ARG_TYPE_String)	certification	Optional
Signature	XML signature over entire contents of the certification element. This is done as specified in [4]. The key used in calculating the signature MUST be the private part of the application-specific key which public part was bound to the attestation of UPnP-Server component. (The public part can be used to verify the signature.) The	certification	Mandatory

Element	Description	Parent	Availability
	Reference element of the XML signature MUST point to the certification element. The SignatureMethod MUST be RSA with SHA1. The KeyInfo element MAY be omitted. The mechanism for generation, exchange and maintainance of keys is out of scope for this specification.		

The elements marked with a (*) can have multiple instances.
The elements marked with a (+) will have implementation specific values which MAY be outside the scope of this specification.

The appUUID MUST be a universally-unique identifier for the application, across application versions and MirrorLink Server platforms. It MUST begin with "uuid:" followed by a 128 bit number that MUST be formatted as specified by the following grammar (taken from [1]):

```
UUID     = 4 * <hexOctet> "-" 2 * <hexOctet> "-"
           2 * <hexOctet> "-" 2 * <hexOctet> "-"
           6 * <hexOctet>
hexOctet = <hexDigit> <hexDigit>
hexDigit = "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"|
           "a"|"b"|"c"|"d"|"e"|"f"|"A"|"B"|"C"|"D"|"E"|"F"
```

The following is an example of an appUUID:

```
"uuid:2fac1234-31f8-11b4-a222-08002b34c003"
```

The entity name MUST be used case-insensitive, when comparing the entry with other values, i.e. the following entries are identical:

1. "CCC" and "CcC"
2. "VW" and "vw"
3. "Volkswagen" and "VolksWagen"

A certificate MAY be valid for only a limited set of localities. The allowed localities, for each driving mode (i.e. restricted or nonRestricted) MUST be listed in the dedicated section, separated by comma. Allowed localities are given below. The 3 letter abbreviations are taken from the IOC country codes.

- "EU" European Union member states
- "EPE" Europe[1] without countries listed separately or EU member states
- "CAN" Canada
- "USA" USA
- "AMERICA" Americas without countries listed separately
- "AUS" Australia
- "KOR" Korea
- "JPN" Japan
- "CHN" China
- "HKG" Hongkong
- "TPE" Taiwan
- "IND" India
- "APAC" APAC states without countries listed separately
- "AFRICA" African countries without countries listed separately
- "WORLD" All countries

The localities MUST be used case-insensitive, when comparing them with other values.

[1] Including Russia and Turkey.

Note: The list of locales, for which a certificate is valid, MUST always include all localities, even if one locale includes other ones.

In order to show that an application has been drive-certified for the entire world, i.e. all localities, the <restricted> entry within A_ARG_TYPE_AppCertificateInfo MUST have the following value:

"EU,EPE,CAN,USA,AMERICA,AUS,KOR,JPN,CHN,HKG,TPE,IND,APAC,AFRICA,WORLD"

A value of "WORLD" is invalid.

2.3 Eventing and Moderation

The following table lists the eventing and moderation properties for each of the service state variables.

Table 2-9: Eventing and Moderation

Variable Name	Evented	Moderated Event	Max. Event Rate	Logical Relation	Min. Delta per Event
AppStatusUpdate	Yes	No	N/A	N/A	N/A
AppListUpdate	Yes	No	N/A	N/A	N/A
A_ARG_TYPE_AppStatus	No	N/A	N/A	N/A	N/A
A_ARG_TYPE_AppID	No	N/A	N/A	N/A	N/A
A_ARG_TYPE_ProfileID	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_AppList	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_Bool	No	N/A	N/A	N/A	N/A
A_ARG_TYPE_INT	No	N/A	N/A	N/A	N/A
A_ARG_TYPE_AppCertificateInfo	No	N/A	N/A	N/A	N/A

2.4 Supporting Multiple Client Profiles

Please refer to the TmClientProfile:1 specifications for more information on support for multiple client profiles and the utilization of profile identifiers (profileIDs) to access parameter settings related to different client profiles. These parameter settings MAY be utilized by the TmApplicationServer service in an implementation-specific manner. For example, the icon preferences provided as part of a client profile can be used to determine the icon format and size specified in the application listing returned by the GetApplicationList action.

During invocation of an action, a MirrorLink UPnP Control Point passes the profileID of a specific client profile to indicate that it would like that profile's parameter settings to be applied during interaction with the MirrorLink UPnPServer device. In this manner, multiple control points MAY have their own custom profiles which govern their interaction with services hosted on the TmServerDevice (such as TmApplicationServer).

Note that the support for working with multiple client profiles is OPTIONAL for MirrorLink UPnP Control Points. Hence, in case a MirrorLink UPnP Control Point does not support multiple client profiles, it MUST set the ProfileID input argument equal to 0, for any actions that it invokes.

Whenever the TmApplicationServer service is accessing and using parameters from a specific client profile, it MUST ensure that it increments the semaphore associated with that profile as specified in the TmClientProfile:1 service specifications. This will ensure that a specific profile cannot be modified while in use in order to avoid any synchronization issues or conflicts. Also, whenever the TmApplicationServer service stops using a client profile, it MUST decrement the semaphore associated with that profile to indicate that it is no longer using that profile.

2.5 Actions

The following table lists the actions supported by the TmApplicationServer service.

Table 2-10: TmApplicationServer Service Actions

Name	Device R/O[1]	Control Point R/O[2]
GetApplicationList	R	R
LaunchApplication	R	R
GetApplicationStatus	R	O
TerminateApplication	R	O
GetApplicationCertificateInfo	R	O
GetCertifiedApplicationsList	R	O
GetAppCertificationStatus	R	O

2.5.1 GetApplicationList

The GetApplicationList action provides a list of applications, which can be launched and terminated remotely. The list includes details such as application name, icons, remoting protocol used, application content category and trust level.

2.5.1.1 Arguments

Table 2-11: Arguments for GetApplicationList

Argument	Direction	relatedStateVariable
AppListingFilter	IN	A_ARG_TYPE_String
ProfileID	IN	A_ARG_TYPE_ProfileID
AppListing	OUT	A_ARG_TYPE_AppList

Parameters:

AppListingFilter (A_ARG_TYPE_String) – Application Listing Filter. This parameter is used by the UPnP Control Point to limit the AppListing value to those applications which meet the filter parameters. It consists of a comma-separated list of A_ARG_TYPE_AppList schema elements, attributes and their values (see Section 3.3for examples). If the value of the AppListingFilter parameter is equal to "*" (default value), all elements and attributes (including OPTIONAL ones, when present) and their values, are returned in AppListing. If the value of the AppListingFilter parameter is equal to "" (empty string), then it is considered to be equivalent to having the value "*".

ProfileID (A_ARG_TYPE_ProfileID) – ProfileID of client profile whose parameter settings will be applied for generating the application list. For example, the icon parameter settings of a specific client profile can be used to generate the `<iconList>` entries in the applicaton list returned by this action.
In case a MirrorLink UPnP Control Point does not use TmClientProfile service it MUST set the ProfileID input argument equal to 0.

Return Value:

AppListing (A_ARG_TYPE_AppList) - Returns a list of applications which are available for remote control and access. The applications listed in AppListing can be controlled using the LaunchApplication, TerminateApplication and GetApplicationStatus actions.

2.5.1.2 Effect on State

None.

2.5.1.3 Errors

Table 2-12: Error Codes for GetApplicationList

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 TmApplicationServer service has rejected the operation.
815	Device Locked	The action cannot be processed as the device hosting the TmApplicationServer service is locked. User needs to unlock the device first.
820	Invalid Argument	The argument passed is invalid.
830	Invalid Profile ID	The profile identifier does not exist or the application cannot use the specified profile identifier.

2.5.2 LaunchApplication

The LaunchApplication action enables a client to remotely start a specific application on the MirrorLink Server device and provide remote access to it if available. Note that this action can be used to launch all types of applications including daemons and other servers which MAY NOT have a UI.

One of the input arguments of this action is the ProfileID of the client profile whose parameter settings will be applied for launching and executing the application. In case the application is a UI application already running using a specific client profile, invoking the LaunchApplication action again using the same ProfileID causes the application to be brought to the foreground and given control of the UI. Hence, this action can be used by the MirrorLink Client to switch between different remote applications in the context of a specific client profile.

The MirrorLink UPnP Server MUST ensure that the implementation of the LaunchApplication action is idempotent in context to a specific client profile. For example, if an application with a specific AppID is already running using a specific client profile, then multiple calls to LaunchApplication using the same AppID and ProfileID will not launch the application again but it will put the application in the foreground. If the application being launched is a UI application, then the MirrorLink Server device MUST give control of the UI to the launched application before returning a response to the LaunchApplication action.

The MirrorLink UPnP Control Point MAY launch a VNC Server directly, if advertised by the MirrorLink UPnP Server. In that case, MirrorLink Client SHOULD receive the MirrorLink Server's current screen content. The MirrorLink Client MAY use LaunchApplication to bring a particular application into the foreground.

The MirrorLink UPnP Control Point MAY launch a BT HFP or BT A2DP component directly, if advertised by the MirrorLink UPnP Server; a timeout MAY happen for LaunchApplication of those components, as user input MAY be needed for pairing and then connecting through BT HFP or BT A2DP. In that case, the MirrorLink Client MAY execute the LaunchApplication action again.

2.5.2.1 Arguments

Table 2-13: Arguments for LaunchApplication

Argument	Direction	relatedStateVariable
AppID	IN	A_ARG_TYPE_AppID
ProfileID	IN	A_ARG_TYPE_ProfileID
AppURI	OUT	A_ARG_TYPE_URI

Parameters:

AppID (A_ARG_TYPE_AppID) – Unique application ID of application to be launched by invocation of this action.

ProfileID (A_ARG_TYPE_ProfileID) – ProfileID of client profile whose parameter settings will be applied for launching and executing the application.
In case a MirrorLink UPnP Control Point does not use TmClientProfile service it MUST set the ProfileID input argument equal to 0.

Return Value:

AppURI – A_ARG_TYPE_URI
This method will return a URI for accessing the remote application using the protocol identifier as given in the <remotingInfo> element of the AppListing returned in response to the GetApplicationList action.

2.5.2.2 Effect on State

This action affects the value of the AppStatusUpdate state variable if it contains an entry corresponding to the value of the AppID argument.

2.5.2.3 Errors

Table 2-14: Error Codes for LaunchApplication

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.
810	Bad AppId	The AppId does not exist or is malformed.
811	Unauthorized AppId	The application identified by this AppId cannot be controlled or accessed remotely.
813	Launch Failed	Failed to launch the application.
814	Resource Busy	The requested application resource is busy, This error can occur when the resource is already busy and resourceStatus in the AppListing is set equal to "NA".
815	Device Locked	The action cannot be processed as the device hosting the TmApplicationServer service is locked. User needs to unlock the device first.
830	Invalid Profile ID	The profile identifier does not exist or the application cannot use the specified profile identifier.

2.5.3 TerminateApplication

The TerminateApplication action enables the client to remotely terminate any application, which is listed in the AppList returned by the GetApplicationList action. It MUST be noted that any application listed as in the AppList can be terminated using this action even if it was not started through invocation of the LaunchApplication action. Furthermore, this TerminateApplication will have no effect on any applications which are not listed in AppList. The MirrorLink UPnP Control Point SHOULD terminate a BT HFP or BT A2DP component only, if the component has been previously launched using LaunchApplication or if the MirrorLink Client provided its Bluetooth MAC address (bdAddr) in A_ARG_TYPE_ClientProfile.

One of the input arguments of this action is the ProfileID of the client profile whose parameter settings are being applied for executing the application. This information is used to ensure that only the instance of the

application which is specific to the client profile is terminated. Applications, which have not been previously launched with LaunchApplication, but are already running, SHOULD be terminated using the default ProfileID 0 (zero).

A MirrorLink Server MAY NOT support terminating an application (i.e. the application is removed from the MirrorLink Server device's process list), but the MirrorLink Server MUST ensure the following behavior in case the MirrorLink has requested the application's termination:

- In case the application to be terminated, is a VNC application in the "`Foreground`", that application MUST change its application status to "`Notrunning`" or "`Background`". The terminated application MUST end any ongoing audio streaming. The MirrorLink Server MAY (potentially launch and) bring another application into the foreground.
- In case the application to be terminated, is a VNC application in the "`Background`", that application MUST change its application status to "`Notrunning`" or stay in "`Background`". The terminated application MUST end any ongoing audio streaming. The MirrorLink Server MUST keep the current foreground applications
- In case the application to be terminated, is a RTP Server, RTP Client, DAP endpoint or CDB Endpoint application, that application MUST change its application status to "`Notrunning`". The MirrorLink Server MUST keep the current foreground applications

If any of the above conditions is fulfilled, the MirrorLink Server MUST respond with "`TerminationResult=true`", otherwise with "`TerminationResult=false`". The MirrorLink Server MUST always notify any application's status change to the MirrorLink Client via the *AppStatusUpdate* evented variable, if that application has been included within the UPnP advertisements.

The MirrorLink UPnP Server MUST ensure that the implementation of the TerminateApplication action is idempotent in the context of a specific client profile. For example, if an application with a specific AppID is not running for a specific client profile, then further calls to TerminateApplication using the same AppID and ProfileID of the client profile will have no effect but will return TerminationResult as True.

The client is NOT REQUIRED to call the TerminateApplication action every time it switches to a different remote application. If a client launches a different remote application without terminating the existing one, the MirrorLink Server MUST either send the original application to the background or it MUST terminate the original application.

2.5.3.1 Arguments

Table 2-15: Arguments for TerminateApplication

Argument	Direction	relatedStateVariable
AppID	IN	A_ARG_TYPE_AppID
ProfileID	IN	A_ARG_TYPE_ProfileID
TerminationResult	OUT	A_ARG_TYPE_Bool

Parameters:

AppID (A_ARG_TYPE_AppID) – Unique application ID of application to be terminated by invocation of this action.

ProfileID (A_ARG_TYPE_ProfileID) – ProfileID of the client profile whose parameter settings are being applied to the application during execution.
In case a MirrorLink UPnP Control Point does not use TmClientProfile service it MUST set the ProfileID input argument equal to 0.

Return Value:

TerminationResult (A_ARG_TYPE_Bool) - Returns true if application terminated successfully, false otherwise.

2.5.3.2 Effect on State

This action affects the value of the AppStatusUpdate state variable if it contains an entry corresponding to the value of the AppID argument.

2.5.3.3 Errors

Table 2-16: Error Codes for TerminateApplication

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.
810	Bad AppId	The AppId does not exist or is malformed.
811	Unauthorized AppId	The application identified by this AppId cannot be controlled or accessed remotely.
815	Device Locked	The action cannot be processed as the device hosting the TmApplicationServer service is locked. User needs to unlock the device first.
830	Invalid Profile ID	The profile identifier does not exist or the application cannot use the specified profile identifier.

2.5.4 GetApplicationStatus

The GetApplicationStatus action provides the current status of one or all applications and allows the client to request automatic updates of status changes for any application listed in the AppList returned by the GetApplicationList action.

2.5.4.1 Arguments

Table 2-17: Arguments for GetApplicationStatus

Argument	Direction	relatedStateVariable
AppID	IN	A_ARG_TYPE_AppID
AppStatus	OUT	A_ARG_TYPE_AppStatus

Parameters:

AppID (A_ARG_TYPE_AppID) - Unique application ID; can be set equal to the string "*" to indicate a wildcard. In case the AppID parameter is set by the client to be equal to "*" then the MirrorLink UPnP Server MUST return the application status of all applications, which can be controlled using the TmApplicationServer service.

Return Value

AppStatus (A_ARG_TYPE_AppStatus) – Status of application(s).

2.5.4.2 Effect on State

None.

2.5.4.3 Errors

Table 2-18: Error Codes for GetApplicationStatus

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.
810	Bad AppId	The AppId does not exist or is malformed.
812	Cannot Determine Status	The status of the specified application(s) cannot be determined.
815	Device Locked	The action cannot be processed as the device hosting the TmApplicationServer service is locked. User needs to unlock the device first.

2.5.5 GetApplicationCertificateInfo

The GetApplicationCertificateInfo action provides certification data for one application.

2.5.5.1 Arguments

Table 2-19: Arguments for GetApplicationCertificateInfo

Argument	Direction	relatedStateVariable
AppID	IN	A_ARG_TYPE_AppID
AppCertification	OUT	A_ARG_TYPE_AppCertificateInfo

Parameters:

AppID (A_ARG_TYPE_AppID) - Unique application ID for the application for which the certificate information SHOULD be returned.

Return Value

AppCertification (A_ARG_TYPE_AppCertificateInfo) – Certificate of application. An empty string is returned, if the application identified by the AppID does not have a valid certificate.

2.5.5.2 Effect on State

None.

2.5.5.3 Errors

Table 2-20: Error Codes for GetApplicationCertificateInfo

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.
810	Bad AppId	The AppId does not exist or is malformed.
815	Device Locked	The action cannot be processed as the device hosting the TmApplicationServer service is locked. User needs to unlock the device first.

2.5.6 GetCertifiedApplicationsList

Get a list of certified applications, matching a set of criteria.

2.5.6.1 Arguments

Table 2-21: Arguments for GetCertifiedApplicationsList

Argument	Direction	relatedStateVariable
AppCertFilter	IN	A_ARG_TYPE_String
ProfileID	IN	A_ARG_TYPE_ProfileID
CertifiedAppList	OUT	A_ARG_TYPE_String

Parameters:

AppCertFilter (A_ARG_TYPE_String) – Application Certification Filter. This parameter is used by the UPnP Control Point to limit the CertifiedAppList to those applications which meet the filter parameters. It consists of a comma-separated list of A_ARG_TYPE_AppCertificateInfo schema elements, attributes and their values (see Section 3.4 for examples). If the value of the AppListingFilter parameter is equal to "" (empty string), then it is considered to be equivalent to having the value "*"(default value).

ProfileID (A_ARG_TYPE_ProfileID) – ProfileID of client profile whose parameter settings will be applied for generating the certified application list. In case a MirrorLink UPnP Control Point does not use TmClientProfile service it MUST set the ProfileID input argument equal to 0.

Return Value

CertifiedAppList (A_ARG_TYPE_String) – Comma separated list of application identifiers of type A_ARG_TYPE_AppId, which are certified based on the giving input filter values. An empty string is returned, if no application matches the input filter criteria.

2.5.6.2 Effect on State

None.

2.5.6.3 Errors

Table 2-22: Error Codes for GetCertifiedApplicationsList

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.
815	Device Locked	The action cannot be processed as the device hosting the TmApplicationServer service is locked. User needs to unlock the device first.
820	Invalid Argument	The argument passed is invalid.
830	Invalid Profile ID	The profile identifier does not exist or the application cannot use the specified profile identifier.

2.5.7 GetAppCertificationStatus

Return the certification status of a given application, under the provided properties.

2.5.7.1 Arguments

Table 2-23: Arguments for GetAppCertificationStatus

Argument	Direction	relatedStateVariable
AppID	IN	A_ARG_TYPE_AppID
AppCertFilter	IN	A_ARG_TYPE_String
ProfileID	IN	A_ARG_TYPE_ProfileID
AppCertified	OUT	A_ARG_TYPE_Bool

Parameters:

AppID (A_ARG_TYPE_AppID) - Unique application ID for the application, for which the certication status SHOULD be returned;

AppCertFilter (A_ARG_TYPE_String) – Application Listing Filter. This parameter is used by the UPnP Control Point to list the criteria to check the application against. It consists of a comma-separated list of A_ARG_TYPE_AppCertificateInfo schema elements, attributes and their values (see Section 3.4 for examples). If the value of the AppListingFilter parameter is equal to "*" (default value), the application passes this parameter. If the value of the AppListingFilter parameter is equal to "" (empty string), then it is considered to be equivalent to having the value "*".

ProfileID (A_ARG_TYPE_ProfileID) – ProfileID of client profile whose parameter settings will be applied. In case a MirrorLink UPnP Control Point does not use TmClientProfile service it MUST set the ProfileID input argument equal to 0.

Return Value

AppCertified (A_ARG_TYPE_Bool) – Boolean value, whether the application is certified ("`true`") or not (`"false"`);

2.5.7.2 Effect on State

None.

2.5.7.3 Errors

Table 2-24: Error Codes for GetAppCertificationStatus

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.
810	Bad AppId	The AppId does not exist or is malformed.
815	Device Locked	The action cannot be processed as the device hosting the TmApplicationServer service is locked. User needs to unlock the device first.
820	Invalid Argument	The argument passed is invalid.

2.5.8 Relationships Between Actions

Figure 2-1 shows the relationship between service actions invoked using the same client profile. Note that the GetApplicationList action has no relationship with the other actions. Also note that actions invoked using different client profiles will have no relationship to each other.

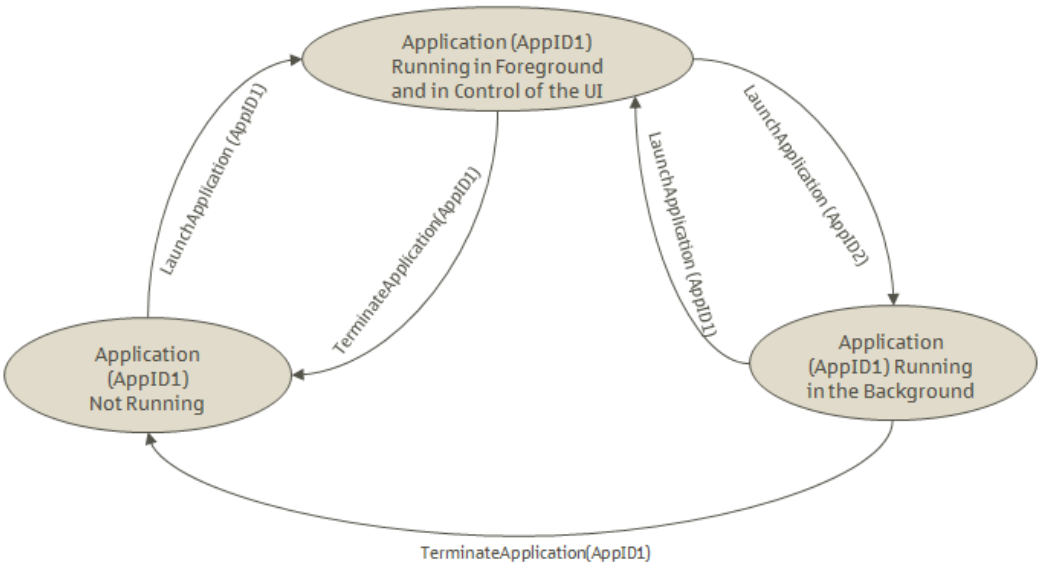

Figure 2-1: Relationship between actions invoked using the same client profile

2.5.9 Error Code Summary

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 2-25: Error Code Summary

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 TmApplicationServer service has rejected the operation. MirrorLink Client SHOULD retry the action.
810	Bad AppId	The AppId does not exist or is malformed. MirrorLink Client SHOULD check the appId (e.g. using GetApplicationList) and retry action. MirrorLink Client SHOULD NOT retry the action with the same appId.
811	Unauthorized AppId	The application identified by this AppId cannot be controlled or accessed remotely. MirrorLink Client SHOULD NOT retry the action.
812	Cannot Determine Status	The status of the specified application(s) cannot be determined. MirrorLink Client SHOULD retry the action.
813	Launch Failed	Failed to launch the application. MirrorLink Client SHOULD retry the action.

ErrorCode	errorDescription	Description
814	Resource Busy	The requested application resource is busy, This error can occur when the resource is already busy and resourceStatus in the AppListing is set equal to "NA". MirrorLink Client SHOULD retry the action. MirrorLink Client SHOULD retry the action only after receiving notification that the resource is becoming free.
815	Device Locked	The action cannot be processed as the device hosting the TmApplicationServer service is locked. User needs to unlock the device first. MirrorLink Client SHOULD NOT retry the action.
820	Invalid Argument	The argument passed is invalid. The MirrorLink Client SHOULD verify the format of the arguments. MirrorLink Client SHOULD NOT retry the action with the same arguments.
830	Invalid Profile ID	The profile identifier does not exist or the application cannot use the specified profile identifier. MirrorLink Client SHOULD check the client profile (GetClientProfile) and its application support from the GetApplicationList response, and retry the action. MirrorLink Client SHOULD NOT retry the action with the same arguments.

Note: 800-899 Error Codes are not permitted for standard actions. See UPnP Device Architecture section on Control for more details.

The MirrorLink Client SHOULD give up after 3 retry attempts. Notification about (finally) failing a UPnP action MAY be necessary. The specification of notification requirements is outside the scope of this specification.

3 Theory of Operation

3.1 Use of Quotation Marks

Throughout the specification, two kinds of quotation marks are used:

- Quotation marks as in "words" are used to highlight a textual element, for readability purpose only. The quotation marks MUST NOT be used within XML schemata, or within arguments of SOAP actions.
- Quotation marks as in "music" are part of the XML or SOAP syntax and MUST be maintained.

Example:

If <protocolId> MUST be "VNC", then

- <protocolId>VNC</protocolId> is valid XML and
- <protocolId>"VNC"</protocolId> is invalid XML.

Note: A MirrorLink 1.0 compliant Client MAY incorrectly include the "" quotation marks within an AppListingFilter parameter. A MirrorLink Server SHOULD ignore those.

3.2 Identification of Applications from the A_ARG_TYPE_AppList

3.2.1 Identifying the VNC Server

If the MirrorLink Server supports a stand-alone VNC Server it MUST set the following entries, so that it can be identified within the A_ARG_TYPE_AppList response to a GetApplicationList() action from the MirrorLink Client:

- <protocolId> MUST be "VNC"
- <appCategory> MUST be "**0xF0000001**" (Server functionality)

The MirrorLink server MUST NOT advertise more than one stand-alone VNC server service for each framebuffer. If the MirrorLink server has multiple framebuffers, the corresponding VNC servers MUST be advertised in dimension order, i.e. framebuffers with bigger dimension appear first.

3.2.2 Identifying Remote VNC based Applications

The MirrorLink Server MUST set the following entries for all VNC based remote applications, so that they can be identified within the A_ARG_TYPE_AppList response to a GetApplicationList() action from the MirrorLink Client:

- <protocolId> MUST be "VNC"

3.2.3 Identifying Audio Links

The MirrorLink Server MUST set the following entries for remote components, providing an audio link, so that they can be identified within the A_ARG_TYPE_AppList response to a GetApplicationList() action from the MirrorLink Client:

The following elements within A_ARG_TYPE_AppList MUST be provided:

<table>
<tr><th></th><th><protocolID></th><th><appCategory></th><th><audioType></th><th><direction></th></tr>
<tr><td>RTP Server</td><td rowspan="2">"RTP"</td><td>"0xF0000001"</td><td rowspan="2">"phone"[1], "application" or "all"</td><td>"out"</td></tr>
<tr><td>RTP Client</td><td>"0xF0000002"</td><td>"in"</td></tr>
<tr><td>BT HFP</td><td>"BTHFP"</td><td rowspan="2">-</td><td>"phone"</td><td>"bi"</td></tr>
<tr><td>BT A2DP</td><td>"BTA2DP"</td><td>"application"</td><td>"out"</td></tr>
</table>

[1] Phone audio includes Voice and Command Control audio, besides cellular call audio.

All other OPTIONAL elements MAY be omitted.

Note: MirrorLink 1.0 Server device MAY omit the audioType entry for BT HFP, BT A2DP and RTP Clients and Servers. Therefore BT HFP MUST be considered as <audioType> being equal to "phone" and BT A2DP as <audioType> being equal to "application". For RTP Clients and Servers the <audioType> MUST be considered being equal to "application".

Note: MirrorLink 1.0 Server device MAY omit the direction entry as well as the appCategory entry. To safely distinguish between an RTP Server and Client, consider the direction's entry default value "out". The MirrorLink server MUST NOT advertise multiple RTP servers or clients with different RTP payload types. The MirrorLink server MUST NOT advertise more than one BT HFP and BT A2DP component. The MirrorLink Server MAY advertise RTP Server or Clients, supporting RTP payload types not being supported from the MirrorLink Client. The MirrorLink Client MUST launch an RTP server or RTP client, which is advertising support for at least one RTP payload type supported from the MirrorLink Client's RTP counterpart.

The A_ARG_TYPE_AppList audioInfo entry provides information, which audio related services are available from the MirrorLink Server.

- **Phone Call (over RTP)**: The advertised RTP Client **and** RTP Server MUST have an audioType of "phone" or "all" **and** MUST include the "Phone Audio" flag within the audioInfo@contentCategory.
- **Phone Call (over BT)**: The advertised BT HFP component MUST have an audioType of "phone" **and** MUST include the "Phone Audio" flag within the audioInfo@contentCategory.
- **Voice Command (over RTP)**: The advertised RTP Client MUST have an audioType of "phone" or "all" **and** MUST include the "Voice Command In" flag within the audioInfo@contentCategory.
- **Voice Command (over BT)**: The advertised BT HFP component MUST have an audioType of "phone" **and** MUST include the "Voice Command In" flag within the audioInfo@contentCategory.
- **Media Streaming (over RTP)**: The advertised RTP Server MUST have an audioType of "application" or "all" **and** MUST include the "Media Audio Out" flag within the audioInfo@contentCategory.
- **Media Streaming (over BT)**: The advertised BT A2DP component MUST have an audioType of "application" **and** MUST include the "Media Audio Out" flag within the audioInfo@contentCategory.

In case a Bluetooth component's resource status is marked as "busy" or "NA" in the A_ARG_TYPE_AppList response, the MirrorLink Client SHOULD subscribe to the AppListUpdate status variable to receive the information, when the resource becomes "free".

3.2.4 Identifying Common Data Bus

If the MirrorLink Server supports the Common Data Bus, it MUST include the Common Data Bus endpoint as an application within A_ARG_TYPE_AppList. In that list it MUST set the following entries, so that the Common Data Bus endpoint can be identified from the MirrorLink client :

The MirrorLink server MUST set the following entries to the designated values:

- <protocolId> MUST be "CDB"
- <appCategory> MUST be "0xF0000000" or non-existing

All other OPTIONAL elements MAY be omitted.

The MirrorLink server MUST NOT advertise more than one Common Data Bus endpoint with the same major version.

3.2.5 Identifying Device Attestation Protocol Server

If the MirrorLink Server supports the Device Attestation Protocol, it MUST include the Device Attestation Protocol Server as an application within A_ARG_TYPE_AppList. In that list it MUST set the following entries, so that the Device Attestation Protocol Server can be identified from the MirrorLink Client:

- <protocolId> MUST be "DAP"
- <appCategory> MUST be "0xF0000001" (Server functionality)

All other OPTIONAL elements MAY be omitted.

The MirrorLink server MUST NOT advertise more than one Device Attestation Protocol server with the same version number.

3.3 Example Values of AppListingFilter

The GetApplicationList action returns application listings that are a match for the parameters specified in *AppListingFilter*.

AppListingFilter is composed of a comma-separated list of *A_ARG_TYPE_AppList* schema elements, attributes and their values. For clarification, the commas in the filter expression are equivalent to an AND operation. For example, "name="music",description="*"" is interpreted as the condition "name="music"" AND "description="*"". Optional as well as REQUIRED schema elements can be included in the AppListingFilter.

To identify an element in the XML hierarchy, the filter SHOULD use the shortest path in the hierarchy, which uniquely identifies the element, according to the following Extended Backus-Naur Form (EBNF):

```
element = [ { parent-element , "@" } ] , child-element;
```

The symbol "@" hereby acts as the delimiter between parent and child elements.

Example 1: The following *AppListingFilter* causes only those application listings with "music" in the value of the REQUIRED <name> element to be returned in the *AppListing* output argument of the GetApplicationList action. 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. The wild-card "*" character MAY be used for non xs:string types as well. In that case it MUST be the only entry.

Example 2: The following *AppListingFilter* argument causes only those application listings with 'png' somewhere in the value of the <mimetype> child element of <icon> to be returned:

```
"icon@mimetype="*png*""
```

Example 3: The following *AppListingFilter* argument causes only those application listings with "platformA" in one of comma-separated entries within the <variant> element to be returned:

```
"variant="platformA""
```

The filter MUST be applied to individual list element, and not to the entire list as a whole; i.e. application will be included, if the <variant> entry equals "platformA,platformB", "platformA", or "platform0,platformA,platformB". A wildcard, will apply only for an individual entry within the comma-separated list. Leading or trailing white spaces MUST NOT be considered, i.e. the application will be included, if the variant entry equals " platformA , platformB ".

The default value of *AppListingFilter* MUST be set equal to "*". If the *AppListingFilter* parameter is equal to "*", all elements and attributes (including OPTIONAL ones, when present) and their values, are returned in *AppListing*.

Elements and attributes REQUIRED by the *A_ARG_TYPE_AppList* schema to be present in the *AppListing* output argument are always returned. Optional elements and attributes MUST be returned, regardless of their values, if the AppListingFilter explicitly specifies the OPTIONAL element or attribute to be returned (as shown in Example 1).

The values of schema elements and attributes specified in *AppListingFilter* are not considered to be case-sensitive for purposes of matching. For example, an AppListingFilter equal to `"name="music""` will return listings of applications which have `"Music"`, `"mUSic"` or `"music"` in their names.

3.4 Example Values of AppCertFilter

The GetCertifiedApplicationsList action returns a list of application that are certified with respect to the parameter specified in the *AppCertFilter*.

AppCertFilter is composed of a comma-separated list of *A_ARG_TYPE_AppCertificateInfo* schema elements, attributes and their values. For clarification, the commas in the filter expression are equivalent to an AND operation. For example, `"name="CCC",restricted="*""` is interpreted as the condition `"name="CCC""` AND `"restricted="*""`. Optional as well as REQUIRED schema elements can be included in the AppCertFilter.

To identify an element in the XML hierarchy, the filter SHOULD use the shortest path in the hierarchy, which uniquely identifies the element, according to the following Extended Backus-Naur Form (EBNF):

```
element = [ { parent-element , "@" } ] , child-element;
```

The symbol "@" hereby acts as the delimiter between parent and child elements.

Example 1: The following *AppCertFilter* causes only those application listings with `"CCC"` in the REQUIRED <entity> element to be returned in the *CertifiedAppList* output argument of the GetCertifiedApplicationsList action:

`"name="CCC""`

The *AppCertFilter* MAY include a wild-card `"*"` character, which generates a match on an unlimited number of UTF-8 characters.

Example 2: The following *AppCertFilter* argument causes only those application listings with `"OEM-A"` in the REQUIRED <entity> element and `"2011"` in the value of the REQUIRED <target> child element of <targeList> to be returned:

`"name="OEM-A",targetList@target="*2011*""`

For clarification: The returned list MUST include only those applications, for which the target `"2011"` is in the target list of the same entity, which has the entity name `"OEM-A"`.

Example 3: The following *AppCertFilter* argument causes only those application listings with `"US"` in one of comma-separated entries within the REQUIRED <restricted> element to be returned:

`"restricted="US""`

The filter MUST be applied to individual list element, and not to the entire list as a whole; i.e. application will be included, if the <restricted> entry equals `"US,EPE"`, `"US"`, or `"EPE,US,JPN"`. A wildcard, will apply only for an individual entry within the comma-separated list. Leading or trailing white spaces MUST NOT be considered, i.e. the application will be included, if the variant entry equals `" US , EPE "`.

The default value of *AppCertFilter* MUST be set equal to `"*"`.

The values of schema elements and attributes specified in *AppCertFilter* are not considered to be case-sensitive for purposes of matching.

3.5 Example Values of State Variables

3.5.1 AppStatusUpdate

The AppStatusUpdate state variable consists of a comma-separated list of appIDs identifying applications whose status has changed. Each entry in the list is of the type A_ARG_TYPE_AppID.

AppStatusUpdate value will consist of a comma separated list of all application identifiers (appIDs) of applications listed in A_ARG_TYPE_AppList when the event is issued by the TmApplicationServer service for the first time.

Example: "0xef128320, 0xffff8320,0x12f128320,0x00128320"

3.5.2 AppListUpdate

The AppListUpdate state variable consists of a comma-separated list of appIDs identifying applications whose entries in the application list have changed. Each entry in the list is of the type A_ARG_TYPE_AppID.

AppListUpdate value will consist of a comma separated list of all application identifiers (appIDs) of applications listed in A_ARG_TYPE_AppList when the event is issued by the TmApplicationServer service for the first time.

Example: "0xffff8320,0xef128320, 0x00128320"

3.5.3 A_ARG_TYPE_AppStatus

The value of the non-evented state variable *A_ARG_TYPE_AppStatus* is an XML block corresponding to a list of applications whose status updates have been requested by the control point. The following examples illustrate some possible values of A_ARG_TYPE_AppStatus.

Example 1: If the application status of a single application with AppID "App_1" is returned.

```
<?xml version="1.0" encoding="UTF-8"?>
<appStatusList>
      <appStatus>
         <appID>0x01</appID>
         <status>
            <profileID>2</profileID>
            <statusType>Foreground</statusType>
         </status>
      </appStatus>
</appStatusList>
```

Example 2: If the application status of multiple applications is returned, for example when AppID is set equal to "*" during invocation of the GetApplicationStatus action.

```
<?xml version="1.0" encoding="UTF-8"?>
<appStatusList>
      <appStatus>
         <appID>0x01</appID>
         <status>
            <profileID>2</profileID>
            <statusType>Foreground</statusType>
         </status>
      </appStatus>
      <appStatus>
         <appID>0x02</appID>
         <status>
            <profileID>2</profileID>
            <statusType>Background</statusType>
         </status>
         <status>
            <profileID>3</profileID>
```

```
            <statusType>Foreground</statusType>
         </status>
      </appStatus>
      <appStatus>
         <appID>0x03</appID>
         <status>
            <profileID>2</profileID>
            <statusType>Notrunning</statusType>
         </status>
      </appStatus>
</appStatusList>
```

3.5.4 A_ARG_TYPE_AppList

The value of the non-evented state variable *A_ARG_TYPE_AppList* is an XML block which lists all the applications which can be remotely controlled by this service. It is returned in response to a GetApplicationList action.

```
<?xml version="1.0" encoding="UTF-8"?>
<appList xml:id="mlServerAppList">
   <app>
      <appID>0x5678</appID>
      <name>Navigation</name>
      <variant>AcmeCar,BetaCar</variant>
      <providerName>Nokia</providerName>
      <providerURL>http://www.nokia.com/maps</providerURL>
      <description>Mobile Navigation Application</description>
      <iconList>
         <icon>
            <mimetype>image/png</mimetype>
            <width>40</width>
            <height>60</height>
            <depth>24</depth>
            <url>/icons/icon5678.png</url>
         </icon>
      </iconList>
      <allowedProfileIDs>1,2,3,4</allowedProfileIDs>
      <remotingInfo>
         <protocolID>VNC</protocolID>
      </remotingInfo>
      <appCertificateURL>
         http://192.168.100.1/navApp.cert
      </appCertificateURL>
      <appInfo>
         <appCategory>0x00050000</appCategory>
         <trustLevel>0x0080</trustLevel>
      </appInfo>
      <displayInfo>
         <contentCategory>0x00000028</contentCategory>
         <contentRules>0x0000000F</contentRules>
         <trustLevel>0x0080</trustLevel>
      </displayInfo>
   </app>
   <app>
      <appID>0x9012</appID>
      <name>RTP Server</name>
      <providerName>Nokia</providerName>
      <providerURL>http://www.nokia.com</providerURL>
      <description>RTP Audio Server</description>
```

```
      <remotingInfo>
         <protocolID>RTP</protocolID>
         <format>99</format>
         <direction>out</direction>
      </remotingInfo>
      <appInfo>
         <appCategory>0xF0000001</appCategory>
         <trustLevel>0x0080</trustLevel>
      </appInfo>
      <audioInfo>
         <audioType>application</audioType>
         <contentCategory>0x02</contentCategory>
      </audioInfo>
      <resourceStatus>free</resourceStatus>
   </app>
   <app>
      <appID>0x9013</appID>
      <name>Bluetooth A2DP</name>
      <description>Bluetooth A2DP Audio Server</description>
      <allowedProfileIDs>1,4</allowedProfileIDs>
      <remotingInfo>
         <protocolID>BTA2DP</protocolID>
         <direction>out</direction>
      </remotingInfo>
      <audioInfo>
         <audioType>application</audioType>
         <contentCategory>0x02</contentCategory>
      </audioInfo>
      <resourceStatus>free</resourceStatus>
   </app>
   <app>
      <appID>0x9014</appID>
      <name>Bluetooth HFP</name>
      <description>Bluetooth HFP Audio </description>
      <allowedProfileIDs>1</allowedProfileIDs>
      <remotingInfo>
         <protocolID>BTHFP</protocolID>
         <direction>bi</direction>
      </remotingInfo>
      <audioInfo>
         <audioType>phone</audioType>
         <contentCategory>0x01</contentCategory>
      </audioInfo>
      <resourceStatus>busy</resourceStatus>
   </app>
   <app>
      <appID>0x8011</appID>
      <name>CDB</name>
      <providerName>Nokia</providerName>
      <description>CDB Server Endpoint</description>
      <remotingInfo>
         <protocolID>CDB</protocolID>
         <format>1.1</format>
      </remotingInfo>
   </app>
   <app>
      <appID>0x9016</appID>
      <name>Device Attestation</name>
```

```
      <remotingInfo>
         <protocolID>DAP</protocolID>
         <format>1.1</format>
      </remotingInfo>
   </app>
   <Signature Id= "AppListSignature"
   xmlns="http://www.w3.org/2000/09/xmldsig#">
      <SignedInfo>
         <CanonicalizationMethod
         Algorithm="http://www.w3.org/2006/12/xml-c14n11"/>
         <SignatureMethod Algorithm=
         "http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
         <Reference URI="#mlServerAppList">
            <Transforms>
               <Transform Algorithm=
               "http://www.w3.org/2006/12/xml-c14n11"/>
            </Transforms>
            <DigestMethod Algorithm=
            "http://www.w3.org/2000/09/xmldsig#sha1"/>
            <DigestValue>
               dGhpcyBpcyBub3QgYSB
               zaWduYXR1cmUK...
            </DigestValue>
         </Reference>
      </SignedInfo>
      <SignatureValue>...</SignatureValue>
      <KeyInfo>
         <KeyValue>
            <DSAKeyValue>
               <P>...</P><Q>...</Q><G>...
               </G><Y>...</Y>
            </DSAKeyValue>
         </KeyValue>
      </KeyInfo>
   </Signature>
</appList>
```

3.6 XML Signature Minimum Set

The MirrorLink Server SHOULD sign the ARG_TYPE_AppListe XML.

The MirrorLink Server MUST sign the ARG_TYPE_AppCertificateInfo XML.

Signatures MUST follow W3C's recommendation on XML signing, as specified in [4]. The W3C recommendation contains many optional elements for handling the different aspects of the XML signing. In order to reduce the complexity, the following requirements MUST be followed for MirrorLink Server and Client devices.

- The **Reference URI** MUST NOT be outside document. It MUST refer to the `<appList>` element, which is the parent of the `<ds:Signature>` element, for the UPnP Application List description. It MUST refer to the `<certificate>` element, which is the parent of the `<ds:Signature>` element, for the UPnP Application Certification Information description. When the URI attribute is omitted, empty or of an unknown format for the MirrorLink Client to recognize, the MirrorLink Client MUST refer to the element listed above.

- MirrorLink Server MUST NOT use XPath or XSLT **XML transformations**

- The MirrorLink Server MUST use **Canonical method** XML version 1.0 (xml-c14n) or 1.1 (xml-c14n11); Canonical XML version 2.0 or later MUST NOT be used.

- The MirrorLink Server MUST use SHA-1 **digest method**; other digest methods MUST NOT be used.
- The MirrorLink Server MUST use RSA-SHA1 **signature method**; other signature methods, like HMAC-SHA1 or DSA-SHA1, MUST NOT be used
- The MirrorLink Client MUST NOT use the **KeyInfo** element to identify a public key to verify the signature; it MUST use the `applicationPublicKey` element obtained from the DAP `attestationResponse`, for the `TerminalMode:UPnP-Server` component instead.

3.7 Handling of Applications Available via Home Screen Application

The MirrorLink Server MAY use the `resourceStatus` entry of "`NA`" within the A_ARG_TYPE_AppList return value to indicate that particular applications with `protocolID` value "`VNC`" are accessible as well, via specific Home Screen Applications.

The following requirements apply to the MirrorLink Server:

- The Home Screen Application, which gives access to other applications, MUST be advertised within the UPnP listings.
- The Home Screen Application MUST be CCC-drive certified, if any of the applications, which are made available, is CCC-drive certified as well.
- The Home Screen Application MUST be CCC-base certified, if any of the applications, which are made available, is CCC-base certified as well.
- The Home Screen Application MUST have `resourceStatus="free"`.
- In Drive mode, the Home Screen Application MUST NOT make any non-drive certified application accessible[1]; Member-drive certified applications MUST be only included, if the MirrorLink Client matches the certifying entity.
- In Drive mode, the Home Screen Application MAY consider MirrorLink Client AppCertFilter settings, for the `restricted/nonRestricted` entries, to not make an application available, but only based on locale information.
- A Mirrorlink Server MUST NOT list any Member-certified application with a non-empty `targetList` entry with a `resourceStatus="NA"`.

The following requirements apply to the MirrorLink Client:

- The MirrorLink Client SHOULD NOT list and allow launching applications from its native screen, which are flagged to included within a separate Home Screen application.

[1] The MirrorLink Server MAY NOT show the application within the Home Screen, or MAY mark the application as not-accessible.

4 XSD Schema

4.1 A_ARG_TYPE_AppStatus XSD Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="urn:schemas-upnp-org:tmapplicationserver:appstatus-1-
0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified"
id="appstatus">
<xs:element name="appStatusList">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="appStatus" minOccurs="1" maxOccurs="unbounded">
    <xs:complexType>
     <xs:sequence maxOccurs="unbounded">
      <xs:element name="appID">
       <xs:simpleType>
        <xs:restriction base="xs:string">
         <xs:pattern value="0[Xx][A-Fa-f0-9]{1,8}"/>
        </xs:restriction>
       </xs:simpleType>
      </xs:element>
      <xs:element name="status" minOccurs="1" maxOccurs="unbounded">
       <xs:complexType>
        <xs:sequence>
         <xs:element name="profileID">
          <xs:simpleType>
           <xs:restriction base="xs:nonNegativeInteger">
            <xs:minInclusive value="0"/>
           </xs:restriction>
          </xs:simpleType>
         </xs:element>
         <xs:element name="statusType">
          <xs:simpleType>
           <xs:restriction base="xs:string">
            <xs:enumeration value="Foreground"/>
            <xs:enumeration value="Background"/>
            <xs:enumeration value="Notrunning"/>
           </xs:restriction>
          </xs:simpleType>
         </xs:element>
         <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
         processContents="lax"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##any" processContents="lax"/>
       </xs:complexType>
      </xs:element>
      <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"
      processContents="lax"/>
     </xs:sequence>
     <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:complexType>
   </xs:element>
   <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"
   processContents="lax"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
 </xs:complexType>
```

```
</xs:element>
</xs:schema>
```

4.2 A_ARG_TYPE_AppList XSD Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="urn:schemas-upnp-org:tmapplicationserver:applist-1-0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
elementFormDefault="qualified" attributeFormDefault="unqualified"
id="applist">
<xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-
20020212/xmldsig-core-schema.xsd"/>
<xs:element name="appList">
 <xs:complexType>
  <xs:sequence minOccurs="1" maxOccurs="1">
   <xs:element name="app" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="appID" minOccurs="1" maxOccurs="1">
       <xs:simpleType>
        <xs:restriction base="xs:string">
         <xs:pattern value="0[Xx][A-Fa-f0-9]{1,8}"/>
        </xs:restriction>
       </xs:simpleType>
      </xs:element>
      <xs:element name="name" type="xs:string" minOccurs="1"/>
      <xs:element name="variant" type="xs:string" minOccurs="0"/>
      <xs:element name="providerName" type="xs:string"
      minOccurs="0"/>
      <xs:element name="providerURL" type="xs:string" minOccurs="0"/>
      <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"
            minOccurs="1" maxOccurs="1"/>
            <xs:element name="width" type="xs:positiveInteger"
            minOccurs="1" maxOccurs="1"/>
            <xs:element name="height" type="xs:positiveInteger"
            minOccurs="1" maxOccurs="1"/>
            <xs:element name="depth" type="xs:positiveInteger"
            minOccurs="1" maxOccurs="1"/>
            <xs:element name="url" type="xs:string"
            minOccurs="1" maxOccurs="1"/>
            <xs:any namespace="##any" minOccurs="0"
            maxOccurs="unbounded" processContents="lax"/>
           </xs:sequence>
           <xs:anyAttribute namespace="##any" processContents="lax"/>
          </xs:complexType>
         </xs:element>
         <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"
         processContents="lax"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##any" processContents="lax"/>
       </xs:complexType>
      </xs:element>
```

```
1         <xs:element name="allowedProfileIDs" type="xs:string"
2         minOccurs="0" default="0"/>
3         <xs:element name="remotingInfo" minOccurs="1" maxOccurs="1">
4          <xs:complexType>
5           <xs:sequence>
6            <xs:element name="protocolID" minOccurs="1" maxOccurs="1">
7             <xs:simpleType>
8              <xs:restriction base="xs:string">
9               <xs:pattern value="(VNC|RTP|BTA2DP|BTHFP|DAP|CDB|
10              NONE|.*-.*)"/>
11             </xs:restriction>
12            </xs:simpleType>
13           </xs:element>
14           <xs:element name="format" type="xs:string" minOccurs="0"
15           maxOccurs="1"/>
16           <xs:element name="direction" minOccurs="0" maxOccurs="1"
17           default="out">
18            <xs:simpleType>
19             <xs:restriction base="xs:string">
20              <xs:enumeration value="out"/>
21              <xs:enumeration value="in"/>
22              <xs:enumeration value="bi"/>
23             </xs:restriction>
24            </xs:simpleType>
25           </xs:element>
26           <xs:element name="audioIPL" type="xs:positiveInteger"
27           minOccurs="0" default="4800"/>
28           <xs:element name="audioMPL" type="xs:positiveInteger"
29           minOccurs="0" default="9600"/>
30           <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
31           processContents="lax"/>
32          </xs:sequence>
33          <xs:anyAttribute namespace="##any" processContents="lax"/>
34         </xs:complexType>
35        </xs:element>
36        <xs:element name="appCertificateURL" type="xs:string"
37         minOccurs="0"/>
38        <xs:element name="appInfo" minOccurs="0">
39         <xs:complexType>
40          <xs:sequence>
41           <xs:element name="appCategory" minOccurs="0"
42           default="0x00000000">
43            <xs:simpleType>
44             <xs:restriction base="xs:string">
45              <xs:pattern value="0[Xx][A-Fa-f0-9]{1,8}"/>
46             </xs:restriction>
47            </xs:simpleType>
48           </xs:element>
49           <xs:element name="trustLevel" minOccurs="0"
50           default="0x0000">
51            <xs:simpleType>
52             <xs:restriction base="xs:string">
53              <xs:pattern value="0[Xx][A-Fa-f0-9]{1,4}"/>
54             </xs:restriction>
55            </xs:simpleType>
56           </xs:element>
57           <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
58           processContents="lax"/>
```

```
          </xs:sequence>
          <xs:anyAttribute namespace="##any" processContents="lax"/>
         </xs:complexType>
        </xs:element>
        <xs:element name="displayInfo" minOccurs="0">
         <xs:complexType>
          <xs:sequence>
           <xs:element name="contentCategory" minOccurs="0"
           default="0x00000000">
            <xs:simpleType>
             <xs:restriction base="xs:string">
              <xs:pattern value="0[Xx][A-Fa-f0-9]{1,8}"/>
             </xs:restriction>
            </xs:simpleType>
           </xs:element>
           <xs:element name="contentRules" minOccurs="0"
           default="0x00000000">
            <xs:simpleType>
             <xs:restriction base="xs:string">
              <xs:pattern value="0[Xx][A-Fa-f0-9]{1,8}"/>
             </xs:restriction>
            </xs:simpleType>
           </xs:element>
           <xs:element name="orientation" minOccurs="0" default="both">
            <xs:simpleType>
             <xs:restriction base="xs:string">
              <xs:enumeration value="landscape"/>
              <xs:enumeration value="portrait"/>
              <xs:enumeration value="both"/>
              <xs:enumeration value="mixed"/>
             </xs:restriction>
            </xs:simpleType>
           </xs:element>
           <xs:element name="trustLevel" minOccurs="0"
           default="0x0000">
            <xs:simpleType>
             <xs:restriction base="xs:string">
              <xs:pattern value="0[Xx][A-Fa-f0-9]{1,4}"/>
             </xs:restriction>
            </xs:simpleType>
           </xs:element>
           <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
           processContents="lax"/>
          </xs:sequence>
          <xs:anyAttribute namespace="##any" processContents="lax"/>
         </xs:complexType>
        </xs:element>
        <xs:element name="audioInfo" minOccurs="0">
         <xs:complexType>
          <xs:sequence>
           <xs:element name="audioType" minOccurs="1">
            <xs:simpleType>
             <xs:restriction base="xs:string">
              <xs:enumeration value="phone"/>
              <xs:enumeration value="application"/>
              <xs:enumeration value="all"/>
              <xs:enumeration value="none"/>
             </xs:restriction>
```

```
            </xs:simpleType>
           </xs:element>
           <xs:element name="contentCategory" minOccurs="1">
            <xs:simpleType>
             <xs:restriction base="xs:string">
              <xs:pattern value="0[Xx][A-Fa-f0-9]{1,8}"/>
             </xs:restriction>
            </xs:simpleType>
           </xs:element>
           <xs:element name="contentRules" minOccurs="0"
           default="0x00000000">
            <xs:simpleType>
             <xs:restriction base="xs:string">
              <xs:pattern value="0[Xx][A-Fa-f0-9]{1,8}"/>
             </xs:restriction>
            </xs:simpleType>
           </xs:element>
           <xs:element name="trustLevel" minOccurs="0"
           default="0x0000">
            <xs:simpleType>
             <xs:restriction base="xs:string">
              <xs:pattern value="0[Xx][A-Fa-f0-9]{1,4}"/>
             </xs:restriction>
            </xs:simpleType>
           </xs:element>
           <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
           processContents="lax"/>
          </xs:sequence>
          <xs:anyAttribute namespace="##any" processContents="lax"/>
         </xs:complexType>
        </xs:element>
        <xs:element name="resourceStatus" minOccurs="0" default="free">
         <xs:simpleType>
          <xs:restriction base="xs:string">
           <xs:enumeration value="free"/>
           <xs:enumeration value="busy"/>
           <xs:enumeration value="NA"/>
          </xs:restriction>
         </xs:simpleType>
        </xs:element>
        <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
        processContents="lax"/>
       </xs:sequence>
       <xs:anyAttribute namespace="##any" processContents="lax"/>
      </xs:complexType>
     </xs:element>
     <xs:element name="Signature" type="ds:SignatureType"
     minOccurs="0"/>
     <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
     processContents="lax"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##any" processContents="lax"/>
   </xs:complexType>
</xs:element>
</xs:schema>
```

4.3 A_ARG_TYPE_AppCertificateInfo XSD Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="urn:schemas-upnp-org:tmapplicationserver:appstatus-1-
0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
elementFormDefault="qualified" attributeFormDefault="unqualified" id="
appcertifcateinfo">
<xs:import schemaLocation="xmldsig-core-schema.xsd"
namespace="http://www.w3.org/2000/09/xmldsig#"/>
<xs:element name="certification">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="appID" minOccurs="1" maxOccurs="1">
    <xs:simpleType>
     <xs:restriction base="xs:string">
      <xs:pattern value="0[Xx][A-Fa-f0-9]{1,8}"/>
     </xs:restriction>
    </xs:simpleType>
   </xs:element>
   <xs:element name="nonce" type="xs:string" minOccurs="1"
   maxOccurs="1"/>
   <xs:element name="appUUID" type="xs:string" minOccurs="0"
   maxOccurs="1"/>
   <xs:element name="entity" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="name" type="xs:string" minOccurs="1"/>
      <xs:element name="targetList" minOccurs="0">
       <xs:complexType>
        <xs:sequence>
         <xs:element name="target" type="xs:string" minOccurs="1"
         maxOccurs="unbounded"/>
         <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"
         processContents="lax"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##any" processContents="lax"/>
       </xs:complexType>
      </xs:element>
      <xs:element name="restricted" type="xs:string" minOccurs="1"/>
      <xs:element name="nonRestricted" type="xs:string" minOccurs="1"/>
      <xs:element name="serviceList" minOccurs="0">
       <xs:complexType>
        <xs:sequence>
         <xs:element name="service" type="xs:string" minOccurs="1"
         maxOccurs="unbounded"/>
         <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"
         processContents="lax"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##any" processContents="lax"/>
       </xs:complexType>
      </xs:element>
      <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
      processContents="lax"/>
     </xs:sequence>
     <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:complexType>
   </xs:element>
```

```
    <xs:element name="properties" minOccurs="0" type="xs:string"/>
    <xs:element name="Signature" type="ds:SignatureType"
    minOccurs="1"/>
    <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
    processContents="lax"/>
   </xs:sequence>
   <xs:anyAttribute namespace="##any" processContents="lax"/>
  </xs:complexType>
</xs:element>
</xs:schema>
```

5 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>GetApplicationList</name>
         <argumentList>
            <argument>
               <name>AppListingFilter</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_String
               </relatedStateVariable>
            </argument>
            <argument>
               <name>ProfileID</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_ProfileID
               </relatedStateVariable>
            </argument>
            <argument>
               <name>AppListing</name>
               <direction>out</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_AppList
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>LaunchApplication</name>
         <argumentList>
            <argument>
               <name>AppID</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_AppID
               </relatedStateVariable>
            </argument>
            <argument>
               <name>ProfileID</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_ProfileID
               </relatedStateVariable>
            </argument>
            <argument>
               <name>AppURI</name>
               <direction>out</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_URI
               </relatedStateVariable>
```

```
            </argument>
         </argumentList>
      </action>
      <action>
         <name>TerminateApplication</name>
         <argumentList>
            <argument>
               <name>AppID</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_AppID
               </relatedStateVariable>
            </argument>
            <argument>
               <name>ProfileID</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_ProfileID
               </relatedStateVariable>
            </argument>
            <argument>
               <name>TerminationResult</name>
               <direction>out</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_Bool
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>GetApplicationStatus</name>
         <argumentList>
            <argument>
               <name>AppID</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_AppID
               </relatedStateVariable>
            </argument>
            <argument>
               <name>AppStatus</name>
               <direction>out</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_AppStatus
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>GetApplicationCertificateInfo</name>
         <argumentList>
            <argument>
               <name>AppID</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_AppID
               </relatedStateVariable>
            </argument>
```

```
            <argument>
               <name>AppCertification</name>
               <direction>out</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_AppCertificateInfo
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>GetCertifiedApplicationsList</name>
         <argumentList>
            <argument>
               <name>AppCertFilter</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_String
               </relatedStateVariable>
            </argument>
            <argument>
               <name>ProfileID</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_ProfileID
               </relatedStateVariable>
            </argument>
            <argument>
               <name>CertifiedAppList</name>
               <direction>out</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_String
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>GetAppCertificationStatus</name>
         <argumentList>
            <argument>
               <name>AppID</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_AppID
               </relatedStateVariable>
            </argument>
            <argument>
               <name>AppCertFilter</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_String
               </relatedStateVariable>
            </argument>
            <argument>
               <name>ProfileID</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_ProfileID
               </relatedStateVariable>
```

```
            </argument>
            <argument>
               <name>AppCertified</name>
               <direction>out</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_Bool
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>
   </actionList>
   <serviceStateTable>
      <stateVariable sendEvents="yes">
         <name>AppStatusUpdate</name>
         <dataType>string</dataType>
      </stateVariable>
      <stateVariable sendEvents="yes">
         <name>AppListUpdate</name>
         <dataType>string</dataType>
      </stateVariable>
      <stateVariable sendEvents="no">
         <name>A_ARG_TYPE_AppStatus</name>
         <dataType>string</dataType>
      </stateVariable>
      <stateVariable sendEvents="no">
         <name>A_ARG_TYPE_AppList</name>
         <dataType>string</dataType>
      </stateVariable>
      <stateVariable sendEvents="no">
         <name>A_ARG_TYPE_AppID</name>
         <dataType>string</dataType>
      </stateVariable>
      <stateVariable sendEvents="no">
         <name>A_ARG_TYPE_ProfileID</name>
         <dataType>ui4</dataType>
         <defaultValue>0</defaultValue>
      </stateVariable>
      <stateVariable sendEvents="no">
         <name>A_ARG_TYPE_URI</name>
         <dataType>uri</dataType>
      </stateVariable>
      <stateVariable sendEvents="no">
         <name>A_ARG_TYPE_String</name>
         <dataType>string</dataType>
      </stateVariable>
      <stateVariable sendEvents="no">
         <name>A_ARG_TYPE_Bool</name>
         <dataType>string</dataType>
         <defaultValue>false</defaultValue>
         <allowedValueList>
            <allowedValue>false</allowedValue>
            <allowedValue>true</allowedValue>
         </allowedValueList>
      </stateVariable>
      <stateVariable sendEvents="no">
         <name>A_ARG_TYPE_INT</name>
         <dataType>ui4</dataType>
      </stateVariable>
```

```
      <stateVariable sendEvents="no">
         <name>A_ARG_TYPE_AppCertificateInfo</name>
         <dataType>string</dataType>
      </stateVariable>
   </serviceStateTable>
</scpd>
```

6 Appendix A – Application Context Information

Trust Level

Trust level used in the Context Information Pseudo Encoding is given in the following table

Table 6-1: Trust Level

Value	Description	Trust Level
0x0000	Unknown	No Trust
0x0040	User Configuration	Trust the user The provided data is under control of the user. The user is either directly setting the values or using provided default settings.
0x0060	Self-Registered Application	Trust the application The provided data is under control of the application. The values are defined from the application developer and provided to the VNC and UPnP MirrorLink server via a specific API.
0x0080	Registered Application	Trust the VNC and UPnP MirrorLink server The provided data is under sole control of the VNC and UPnP MirrorLink server. The application is known to them and MUST be uniquely identified. The user and the application MUST NOT be able to change the values.
0x00A0	Application certificate[1]	Trust a 3rd party certification entity The provided data is under sole control of the VNC and UPnP MirrorLink server. The data is derived from a valid application certificate. The VNC and UPnP MirrorLink server MUST NOT change the provided data. The user and the application MUST NOT be able to change the values.

Application Categories

The application categories and sub-categories are given in the following table. The table can be amended in future versions of the specifications.

Table 6-2: Application Categories

Application Category	Application Sub-category	Description
0x0000	0x0000	Unknown Application
0x0001	0x0000	General UI Framework
	0x0001	Home screen / Start-up Screen[2]
	0x0002	Menu
	0x0003	Notification
	0x0004	Application listing
	0x0005	Settings
0x0002	0x0000	General Phone call application
	0x0001	Contact list
	0x0002	Call log

[1] The structure of the application certificate, the certification provider and the certification process are not subject to this specification.

[2] This application category MUST describe the phone´s start-up screen, e.g. shown after booting or after switching to the automotive context.

Application Category	Application Sub-category	Description
0x0003	0x0000	General Media applications
	0x0001	Music
	0x0002	Video
	0x0003	Gaming
	0x0004	Image
0x0004	0x0000	General Messaging applications
	0x0001	SMS
	0x0002	MMS
	0x0003	Email
0x0005	0x0000	General Navigation
0x0006	0x0000	General Browser
	0x0001	Application Store
0x0007	0x0000	General Productivity
	0x0001	Document Viewer
	0x0002	Document Editor
0x0008	0x0000	General Information
	0x0001	News
	0x0002	Weather
	0x0003	Stocks
	0x0004	Travel
	0x0005	Sports
	0x0006	Clock
0x0009	0x0000	General Social Networking
0x000A	0x0000	General Personal Information Management
	0x0001	Calendar
	0x0002	Notes
0xF000	0x0000	General UI less applications
	0x0001	Server functionality (use protocolID[1] to identify the server protocol)
	0x0002	Client functionality (use protocolID to identify the client protocol)
	0x0010	Voice Command Engine (MUST only be used in RTP header extensions)
	0x0020	Conversational Audio (MUST only be used in RTP header extensions)
	0xFFFF	Switch to MirrorLink Client native UI
0xFFFE	0x0000 – 0xFFFF	Testing & Certification The use of this category will be specified in a separate “Testing and Certification” document.
0xFFFF	0x0000	General System
	0x0001	PIN input for Device unlock
	0x0002	Bluetooth PIN code Input

[1] ProtocolID is provided in response to a GetApplicationList action within TmApplicationServer:1 service.

Application Category	Application Sub-category	Description
	0x000F	Other password input
	0x0010	Voice Command Confirmation

Content Categories

The content categories are represented as 32-bit unsigned integer values.

The following table lists content categories for visual content of an application and are specified as values of the contentCategory child element of the displayInfo element in the A_ARG_TYPE_AppList state variable for the TmApplicationServer:1 service.

Table 6-3: Content Categories for Visual Content

Content Category (Bit)	Description
0	Text
1	Video
2	Image
3	Vector Graphics
4	3D Graphics
5	User Interface (e.g. Application menu)
16	Car Mode (Application user interface is complying with all rules for a restricted driving mode)
31	Miscellaneous content

The following table lists content categories for audio content of an application from the MirrorLink server's perspective and are specified as values of the contentCategory child element of the audioInfo element in the A_ARG_TYPE_AppList state variable for the TmApplicationServer:1 service.

Table 6-4: Content Categories for Audio Content

Content Category (Bit)	Description
0	Phone Audio
1	Media Audio Out
2	Media Audio In
3	Voice Command Out
4	Voice Command In
31	Miscellaneous content

If no bit is set, then the content category of that application MUST be treated as "Unknown".

Content Rules

The content rules, which SHOULD be followed from the MirrorLink server's User Interface, in order to minimize driver distraction are given in the following table. Each rule has a unique rule identifier (ruleID) and some of the rules also require an additional value (ruleValue).

Table 6-5: Content Rules

ruleID	Description	ruleValue
0	Minimum font size REQUIRED. ruleValue is given as the minimum font size REQUIRED in pixel divided by the target vertical screen resolution in pixel. MirrorLink server	Mandatory

ruleID	Description	ruleValue
	assumes linear scaling of its provided framebuffer to the target resolution. The fixed-point format MUST be Q0.31[1], represented in hexadecimal format including the hex delimiter “0x”. Negative number MUST not be used.	
1	No video is shown. A video is defined as a sequence of motion images, representing a scene or a sequence of scenes.	Not used
2	No automatic scrolling text . The rule does not apply to user controlled scrolling.	Not used
3	Maximum feedback time ruleValue gives maximum feedback time allowed after user input in [ms]. The format MUST be 32 bit unsigned integer (uint32_t), represented in hexadecimal format.	Mandatory

Note: Currently, only visual content rules are defined. This version of the specification does not define any audio content rules.

[1] Q*m.n* is a fixed-point number format, where the number of fractional (*n*) and integer (*m*) bits are specified. Q0.32 defines fixed-point numbers within the range [-1, 1-2^{-31}] at a resolution of 2^{-31}.

7 References

[1] UPnP Forum, "UPnP Device Architecture 1.0", 24 April 2008, http://www.upnp.org

[2] IETF, RFC 2119, "Keys words for use in RFCs to Indicate Requirement Levels", March 1997. http://www.ietf.org/rfc/rfc2119.txt

[3] IETF, RFC 3986, "Uniform Resource Identifier (URI): Generic Syntax", January 2005, http://tools.ietf.org/html/rfc3986

[4] W3C, "XML Signature Syntax and Processing (Second Edition)", W3C Recommendation, 10 June 2008. http://www.w3.org/TR/xmldsig-core/

[1] For a device this column indicates whether the action MUST be implemented or not, where *R* = REQUIRED, *O* = OPTIONAL, *CR* = CONDITIONALLY REQUIRED, *CO* = CONDITIONALLY OPTIONAL, *X* = Non-standard, add *-D* when deprecated (e.g., *R-D*, *O-D*)..

[2] For a control point this column indicates whether a control point MUST be capable of invoking this action, where *R* = REQUIRED, *O* = OPTIONAL, *CR* = CONDITIONALLY REQUIRED, *CO* = CONDITIONALLY OPTIONAL, *X* = Non-standard, add *-D* when deprecated (e.g., *R-D*, *O-D*).