Car Connectivity Consortium

MirrorLink®

UPnP Notification Server Service

Version 1.1.4
(CCC-TS-028)

VERSION HISTORY

Version	Date	Comment
1.1	31 March 2012	Approved Version
1.1.1	27 September 2012	Approved Errata Version
1.1.2	05 March 2013	Approved Errata Version
1.1.3	29 August 2013	Approved Errata Version
1.1.4	05 November 2013	Approved Errata Version

LIST OF CONTRIBUTORS

Benesch, Matthias (Editor)	Daimler AG
Brakensiek, Jörg	Nokia Corporation
Lee, Sungjin	Samsung Electronics
Kim, Jungwoo (Editor)	LG Electronics
Kim, Kyungguen	LG Electronics
Kim, Mingoo	LG Electronics
Park, Hoyeon (Editor)	Samsung Electronics

LEGAL NOTICE

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

TERMS AND ABBREVIATIONS

ML	MirrorLink
VNC	Virtual Network Computing

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 the *TmNotificationServer* service.

The specification lists a series of requirements, either explicitly or within the text, which are mandatory elements for a 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", mean that the definition is an absolute requirement of the specification.
- MUST NOT: This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification.
- SHOULD: This word, or the adjective "RECOMMENDED", mean 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" mean 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 *TmNotificationServer* service is an UPnP service that allows control points to receive diverse notifications from the devices that support the *TmNotificationServer* service.

The *TmNotificationServer* service enables the following features to:

- send a notification to the head unit
- get an action described in the notification

2 SERVICE MODELING DEFINITION

2.1 Service Type

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

urn:schemas-upnp-org:service:*TmNotificationServer:1*

TmNotificationServer defined in this specification refers to the same service type.

2.2 TmNotificationServer Service Architecture

This service provides the features for a MirrorLink UPnP Control Point to receive notifications from a MirrorLink UPnP Server Device.

Based on information of the notification evented to the MirrorLink UPnP Control Point, the MirrorLink UPnP Control Point MAY launch the applications to bring it to foreground on the MirrorLink Server or MAY create its own native user interface. In both cases, this does allow the end-user to act on the given notification.

2.3 State Variables

Note: For first-time reader, it MAY be more insightful to read the theory of operations first and then the action definitions before reading the state variable definitions.

2.3.1 State Variable Overview

Table 2-1: Service State Variables

Variable Name	Req. or Opt.	Data Type	Allowed Value	Default Value	Eng. Units
ActiveNotiEvent	R	string	Undefined	Empty string	N/A
NotiAppListUpdate	R	string	Undefined	Empty string	N/A
A_ARG_TYPE_Notification	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_ActionID	R	string	Undefined	Empty string	N/A
A_ARG_TYPE_NotiID	R	string	Undefined	Empty string	N/A
A_ARG_TYPE_String	R	string	Undefined	Empty string	N/A
A_ARG_TYPE_URI	R	string	Undefined	Empty string	N/A
A_ARG_TYPE_INT	R	ui4	Undefined	0	N/A
A_ARG_TYPE_Bool	R	string	true | false	false	N/A

[1] *R = REQUIRED, O = OPTIONAL, X = Non-standard*

2.3.2 ActiveNotiEvent

ActiveNotiEvent is an evented state variable of type A_ARG_TYPE_NotiID, which contains the most urgent notification that needs to be handled from the MirrorLink UPnP Control Point. ActiveNotiEvent MUST originate from an application (as given in appID), which has been set using the SetAllowedApplications actions.

It is the responsibility of the MirrorLink UPnP Server to decide on the most urgent notification.

Note: If a notification event A gets overloaded by another notification event B, the notification event A MAY become pending again, once notification event B is cleared and A is still pending. In that case the MirrorLink UPnP Server MUST again provide an event update for the notification event A.

If the state variable is an empty string, no notification is available on the MirrorLink UPnP Server to be handled from the MirrorLink UPnP Control Point.

On receiving an AcitveNotiEvent event, the MirrorLink UPnP Control Point can query specific notification by invoking the GetNotication action.

The MirrorLink UPnP Server MUST clear the active notification if the MirrorLink UPnP Control Point has responded to the notification by either using the InvokeNotiAction() action or by launching the respective application using TmApplicationServer:1 service LaunchApplication() action. The MirrorLink UPnP Server MUST clear the active notification if the notification is not available on the MirrorLink UPnP Server anymore.

When the event is issued the first time, the ActiveNotiEvent value MUST contain either a single value of type A_ARG_TYPE_NotiID, in case a notification is available, or of an empty string, in case no notification is available.

2.3.3 NotiAppListUpdate

NotiAppListUpdate is an evented state variable of type A_ARG_TYPE_String, which contains a comma separated list of applications identifiers of applications, supporting notifications. Each application identifier is of type A_ARG_TYPE_AppID.

The 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 application identifiers of those applications, whose entries in supported applications list have changed since the last time an event notification was sent out (i.e. applications which either have added or removed notification support).

On receiving a NotiAppListUpdate event, a MirrorLink UPnP Control Point can retrieve the supported application list by invoking the GetSupportedApplications action, to validate, whether an application has removed or added notification support.

NotiAppListUpdate value MUST consist of a comma separated list of all application identifiers from applications supporting notification, when the event is issued by the TmNotificationServer service for the first time.

2.3.4 A_ARG_TYPE_Notification

The format of the *A_ARG_TYPE_Notification* state variable is an XML document. It includes detailed information about a notification.

Table 2-2: Structure of the A_ARG_TYPE_Notification

Element	Description	Parent	Availability
notification	Notification element contains detailed information of an event occurred on a phone and is delivered to the MirrorLink UPnP Control Point	-	Required
notiID	Unique identifier of Notification event. (A_ARG_TYPE_NotiID)	notification	Required
notiTitle	Title of the Notification event. In other words, it is a name of an event occurred. For example, a title of the notification “New text message” or “New email” will be showed as a notification pop-up window. MirrorLink UPnP Control Point MUST trim from the right all characters in excess of the specified	notification	Required

	notiTitleMaxLength parameter within the TmClientProfile service. (A_ARG_TYPE_String)		
notiBody	Body of the Notification event. It includes detailed information of an event for a user. For example, text message content for a new text message event, or caller ID for an incoming call event. MirrorLink UPnP Server MAY include white space charaters, like tab, line feed and carriage return. MirrorLink UPnP Control Point MUST trim from the right all characters in excess of the specified notiBodyMaxLength parameter within the TmClientProfile service. (A_ARG_TYPE_String) **Default**: Empty String	notification	Optional
iconList	List of available notification icons	notification	Optional
icon*	Describes an notification icon	iconList	Required
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. • MirrorLink UPnP Control Point 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 to icon (A_ARG_TYPE_URI)	icon	Required
appID	Application ID of the notification to let the MirrorLink UPnP Control Point know where the notification comes from. (A_ARG_TYPE_AppID)	notification	Required
actionList	A list of actions for a notification. The list is provided by an application initiating the notification so a user can directly select one of those actions for the notification. For example, the user can "Reply" to the new text message or "Ignore" it. The list includes "Reply" and "Ignore" actions as its elements.	notification	Optional
action*	Individual action, associated with the notification.	actionList	Required

	MirrorLink UPnP Control Point MUST remove from the end all actions in excess of the specified maxActions parameter within the TmClientProfile service.		
actionID	Unique identifier of an action. When a user selects an action for a notification through the native notification UI served by the MirrorLink UPnP Control Point, actionID MUST be sent to the MirrorLink UPnP Server. MUST be non-zero (0x0000) (A_ARG_TYPE_ActionID)	action	Required
action-Name	Action name. This name will be shown as a button label on the native notification UI. MirrorLink UPnP Control Point MUST trim from the right all characters in excess of the specified actionNameMaxLength parameter within the TmClientProfile service. (A_ARG_TYPE_String)	action	Required
launchApp	Application launch required Launch application after invoked the action, as given in the application ID (<appID>) if value is set to true. (A_ARG_TYPE_Bool) **Default**: false	action	Optional
iconList	List of available action icons	action	Optional
icon*	Describes an action icon	iconList	Required
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. • MirrorLink UPnP Control Point 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 to icon (A_ARG_TYPE_URI)	icon	Required
Signature	XML signature over entire contents of the notification element. This is done as specified in [5]. The key used in calculating the signature MUST be the private part of the application-specific key	notification	Optional

	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 notification 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.

2.3.5 A_ARG_TYPE_AppID

A UTF-8 encoded string representing 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)

2.3.6 A_ARG_TYPE_ProfileID

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

2.3.7 A_ARG_TYPE_ActionID

A UTF-8 encoded string representing an unsigned 32-bit integer in hexadecimal format (with '0x' prefix) which denotes the unique action 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_ActionID 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_ActionID 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)

2.3.8 A_ARG_TYPE_NotiID

A string formatted as UTF-8 representing a notification identifier, which has been provided by the given applications, identified by ApplicationID. The format is given as:

```
NotifiationID@ApplicationID
```

NotificationID is a 32-bit integer in hexadecimal format (with '0x' prefix). ApplicationID is of Type A_ARG_TYPE_AppID.

Valid examples, all referring to the same notification identifier, are given below

- `0x00000001@0x0000000a`
- `0x00000001@0x0000000A`

- `0x01@0x0a`
- `0X01@0X0a`
- `0x1@0xa`
- `0x1@0xA`

The NotificationID MUST be unique within the given application. If an application runs out of not-used NotificationIDs, it MUST NOT send any further notifications.

2.3.9 A_ARG_TYPE_String

A simple string type (UTF-8).

2.3.10 A_ARG_TYPE_URI

A string encoded as UTF-8 representing a URI.

2.3.11 A_ARG_TYPE_INT

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

2.3.12 A_ARG_TYPE_Bool

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

2.4 Eventing and Moderation

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

Table 2-3: Eventing and Moderation

Variable Name	Evented	Moderated Event	Max. Event Rate	Logical Relation	Min. Delta per Event
ActiveNotiEvent	Yes	No	N/A	N/A	N/A
NotiAppListUpdate	Yes	No	N/A	N/A	N/A
A_ARG_TYPE_Notification	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_ActionID	No	N/A	N/A	N/A	N/A
A_ARG_TYPE_NotiID	No	N/A	N/A	N/A	N/A
A_ARG_TYPE_String	No	N/A	N/A	N/A	N/A
A_ARG_TYPE_URI	No	N/A	N/A	N/A	N/A
A_ARG_TYPE_INT	No	N/A	N/A	N/A	N/A
A_ARG_TYPE_Bool	No	N/A	N/A	N/A	N/A

2.5 Supporting Multiple Client Profiles

Please refer to the TmClientProfile:1 specification in [4] 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 TmNotificationServer service in an implementation-specific manner.

The MirrorLink UPnP Control Point selects the profileID to be used from the MirrorLink server when setting the notification configuration in the SetAllowedApplications action. A profile MAY provide information about the potential limits in e.g. number of actions, length of action names etc.

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.

2.6 Actions

Table 2-4: Supported actions

Name	Device R/O	Control Point R/O
GetNotification()	R	O
GetSupportedApplications()	R	O
SetAllowedApplications()	R	R
InvokeNotiAction()	R	O

2.6.1 GetNotification

GetNotification action provides the detailed information of the notification to the MirrorLink UPnP Control Point.

The MirrorLink Client MUST clear any active notification, using `InvokeNotiAction` with `ActionID = 0x00`, if the XML signature within the `Notification` response (A_ARG_TYPE_Notification) is failing validation.

2.6.1.1 Arguments

Table 2-5: Arguments for GetNotification

Argument	Direction	relatedStateVariable
ProfileID	IN	A_ARG_TYPE_ProfileID
NotiID	IN	A_ARG_TYPE_NotilD
Notification	OUT	A_ARG_TYPE_Notification

Parameters:

ProfileID (A_ARG_TYPE_ProfileID) – ProfileID of the client profile whose parameter settings were 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.

NotiID (A_ARG_TYPE_NotilD) – Unique notification ID.

Return Value:

Notification (A_ARG_TYPE_Notification) - Returns the XML formatted notification information.

2.6.1.2 Effect on State

None.

2.6.1.3 Errors

Table 2-6: Error Codes for GetNotification

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 NotiID	The NotiId does not exist or is malformed.
815	Device Locked	The action cannot be processed as the device hosting the TmNotificationServer service is locked. User needs to un-lock the device first.
830	Invalid Profile ID	The profile identifier does not exist or the application cannot use the specified profile identifier.

2.6.2 GetSupportedApplications

The GetSupportedApplications action provides a list of applications supporting a notification.

2.6.2.1 Arguments

Table 2-7: Arguments for GetSupportedApplications

Argument	Direction	relatedStateVariable
ProfileID	IN	A_ARG_TYPE_ProfileID
AppIDs	OUT	A_ARG_TYPE_String

Parameters:

ProfileID (A_ARG_TYPE_ProfileID) – ProfileID of the client profile whose parameter settings were 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:

AppIDs (A_ARG_TYPE_String) – Comma separated list of applications identifiers of applications, supporting notifications. Each application identifier is of type A_ARG_TYPE_AppID.

2.6.2.2 Effect on State

None.

2.6.2.3 Errors

Table 2-8: Error Codes for GetSupportedApplications

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 TmNotificationServer service is locked. User needs to un-lock the device first.

ErrorCode	errorDescription	Description
830	Invalid Profile ID	The profile identifier does not exist or the application cannot use the specified profile identifier.

2.6.3 SetAllowedApplications

The SetAllowedApplications action enables a MirrorLink UPnP Control Point to define the MirrorLink Server applications from which it wants to receive notifications. The MirrorLink UPnP Control Point always provides a complete list of applications, from which it wants to receive notifications, i.e. incremental additions are not supported.

Note: The MirrorLink UPnP Server will provide notifications from applications, which are running on the MirrorLink Server, independent of whether the application has been launched via UPnP TmApplication-Server:1 service LaunchApplication SOAP action or via different means.

2.6.3.1 Arguments

Table 2-9: Arguments for SetAllowedApplications

Argument	Direction	relatedStateVariable
ProfileID	IN	A_ARG_TYPE_ProfileID
AppIDs	IN	A_ARG_TYPE_String

Parameters:

ProfileID (A_ARG_TYPE_ProfileID) – ProfileID of the client profile whose parameter settings were 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.

AppIDs (A_ARG_TYPE_String) – Comma separated list of application IDs the MirrorLink UPnP Control Point would like to receive notifications from. Each application identifier is of type A_ARG_TYPE_AppID. If the value of the AppIDs parameter is equal to "*" (default value), all applications supporting a notification are allowed. An AppIDs parameter value, equal to "" (empty string), defines an empty list, in which case the MirrorLink UPnP Control Point does not want to receive a notification from any application.

Return Value:

None.

2.6.3.2 Effect on State

None.

2.6.3.3 Errors

Table 2-10: Error Codes for SetAllowedApplications

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	One AppID does not exist or is malformed.
820	Invalid Argument	The argument passed is invalid.

ErrorCode	errorDescription	Description
815	Device Locked	The action cannot be processed as the device hosting the TmNotificationServer service is locked. User needs to un-lock the device first.
830	Invalid Profile ID	The profile identifier does not exist or the application cannot use the specified profile identifier.

2.6.4 InvokeNotiAction

InvokeNotiAction action sends the action ID to the MirrorLink UPnP Server.

2.6.4.1 Arguments

Table 2-11: Arguments for InvokeNotiAction

Argument	Direction	relatedStateVariable
ProfileID	IN	A_ARG_TYPE_ProfileID
NotiID	IN	A_ARG_TYPE_NotilD
ActionID	IN	A_ARG_TYPE_ActionID

Parameters:

ProfileID (A_ARG_TYPE_ProfileID) – ProfileID of the client profile whose parameter settings were 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.

NotiID (A_ARG_TYPE_NotilD) – Unique notification ID.

ActionID (A_ARG_TYPE_ActionID) – Unique action ID. The ActionID defines the action the MirrorLink UPnP Server MUST invoked. If ActionID is set to zero (0x0000), the MirrorLink UPnP Server MUST NOT invoke any action. In both cases the active notification is cleared.

Return Value:

None

2.6.4.2 Effect on State

None.

2.6.4.3 Errors

Table 2-12: Error Codes for InvokeNotiAction

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.
816	Action Failed	Failed to invoke action.
815	Device Locked	The action cannot be processed as the device hosting the TmNotificationServer service is locked. User needs to un-lock the device first.

ErrorCode	errorDescription	Description
830	Invalid Profile ID	The profile identifier does not exist or the application cannot use the specified profile identifier.

2.6.5 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 2-13: 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.
810	Bad AppID	One 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.
815	Device Locked	The action cannot be processed as the device hosting the TmNotificationServer service is locked. User needs to un-lock the device first. MirrorLink Client SHOULD NOT retry the action.
816	Action Failed	Failed to invoke 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 Initialization steps

The following sequence describes how to initialize the notification service, and how to set the notification filter.

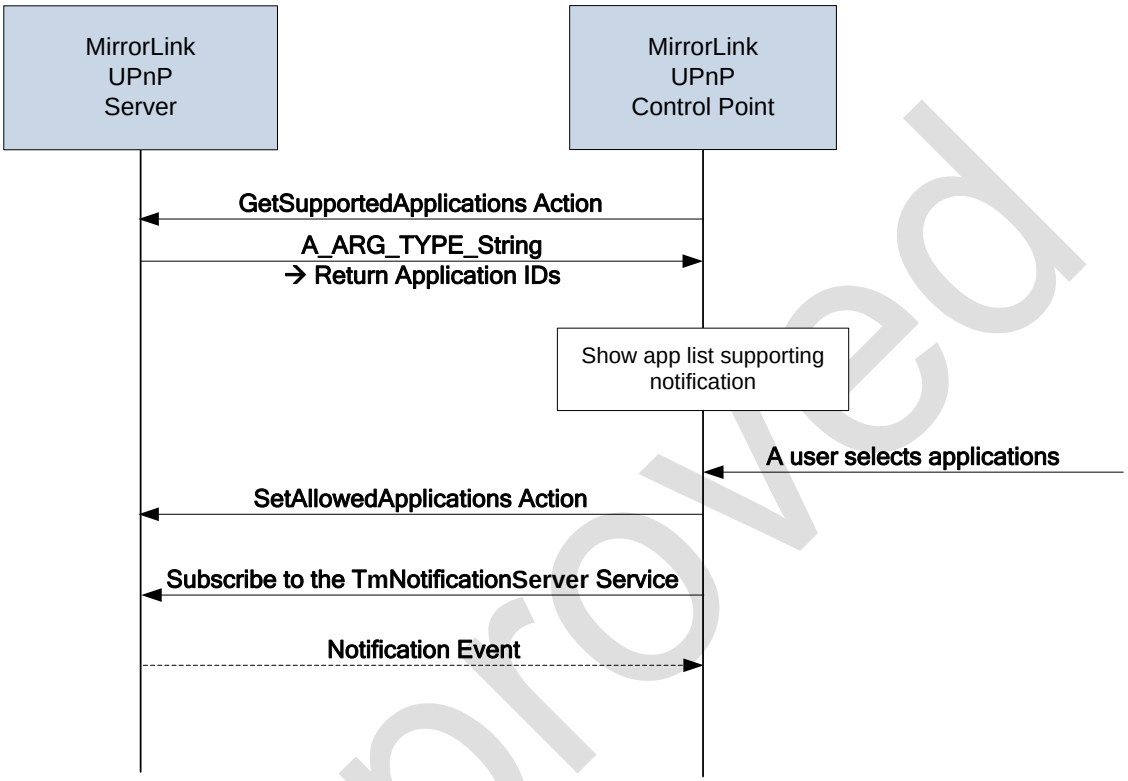

Figure 1: Flow basics for initialization

3.2 Handling of notification

3.2.1 Not using Head Unit UI for notification

The following sequence describes how to handle the notification which comes from the MirrorLink UPnP Server when the MirrorLink UPnP Control Point does not support its own native notification UI. To show a notification UI, VNC SHALL be used.

1. MirrorLink UPnP Server sends the ActiveNotiEvent to the MirrorLink UPnP Control Point.
2. MirrorLink UPnP Control Point launches the respective application based on the ApplicationID information given within the ActiveNotiEvent (NotifiationID@ApplicationID).
3. MirrorLink UPnP Control Point handles the return value of the LaunchApplication action. For example, if the URL in the return value of the LaunchApplication action is “VNC://…”, the MirrorLink UPnP Control Point handles the VNC related process.
4. When a user see the notification UI through VNC, the user directly takes the action (click a button on the screen) for the notification.

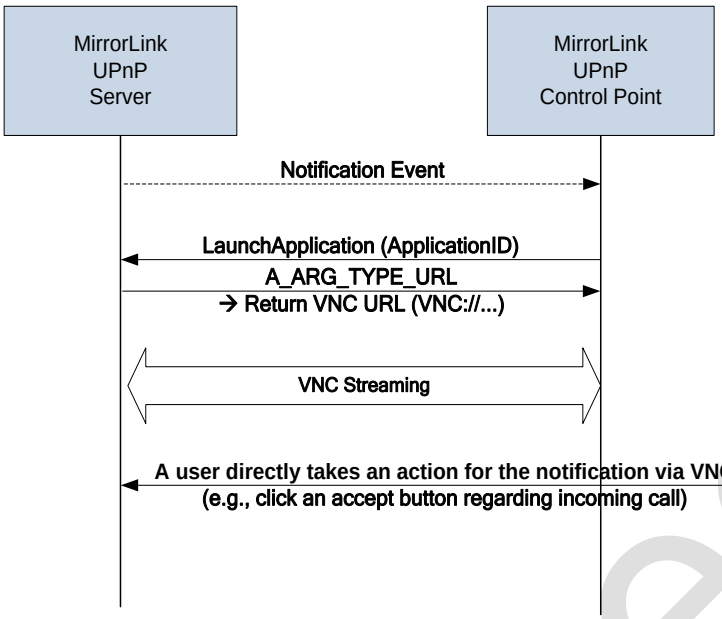

Figure 2: Flow basics not to use MirrorLink Client UI for notification

In case the MirrorLink Client does not launch an application, using LaunchApplication action, it MUST use the `InvokeNotiAction` action with the `actionID = 0x00`, in order to clear the notification, as shown in Figure 3.

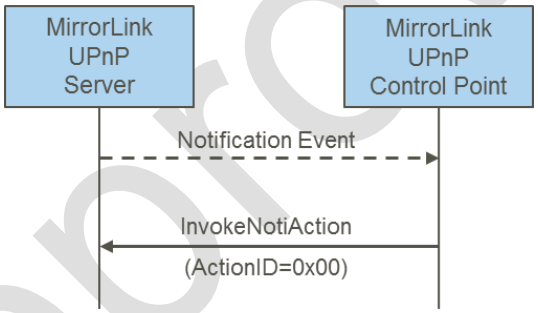

Figure 3: MirrorLink Client immediately clears pending Notification

3.2.2 Using Head Unit UI for notification

The following sequence describes how to handle the notification which comes from the MirrorLink UPnP Server when the MirrorLink UPnP Control Point supports its own native notification UI to show a notification UI.

1. MirrorLink UPnP Server sends the ActiveNotiEvent to the MirrorLink UPnP Control Point.
2. MirrorLink UPnP Control Point invokes GetNotification action with NotiID (A_ARG_TYPE_NotiID) included in the previous ActiveNotiEvent to get detail information of the notification.
3. MirrorLink UPnP Control Point renders of a notification UI based on the return message of GetNotification action received from the MirrorLink UPnP Server.
4. A user clicks one of the buttons on the notification UI.
5. MirrorLink UPnP Control Point invokes the InvokeNotiAction with a <actionID> value.
6. MirrorLink UPnP Control Point launches the respective application having the <appID> value in the Notification information if a <launchApp> value of the selected button by a user has "true"; otherwise the MirrorLink UPnP Control Point will not launch the application.
7. MirrorLink UPnP Control Point handles the return value of the LaunchApplication action. For example, if the URL in the return value of the LaunchApplication action is "VNC://…", the MirrorLink UPnP Control Point handles the VNC related process.

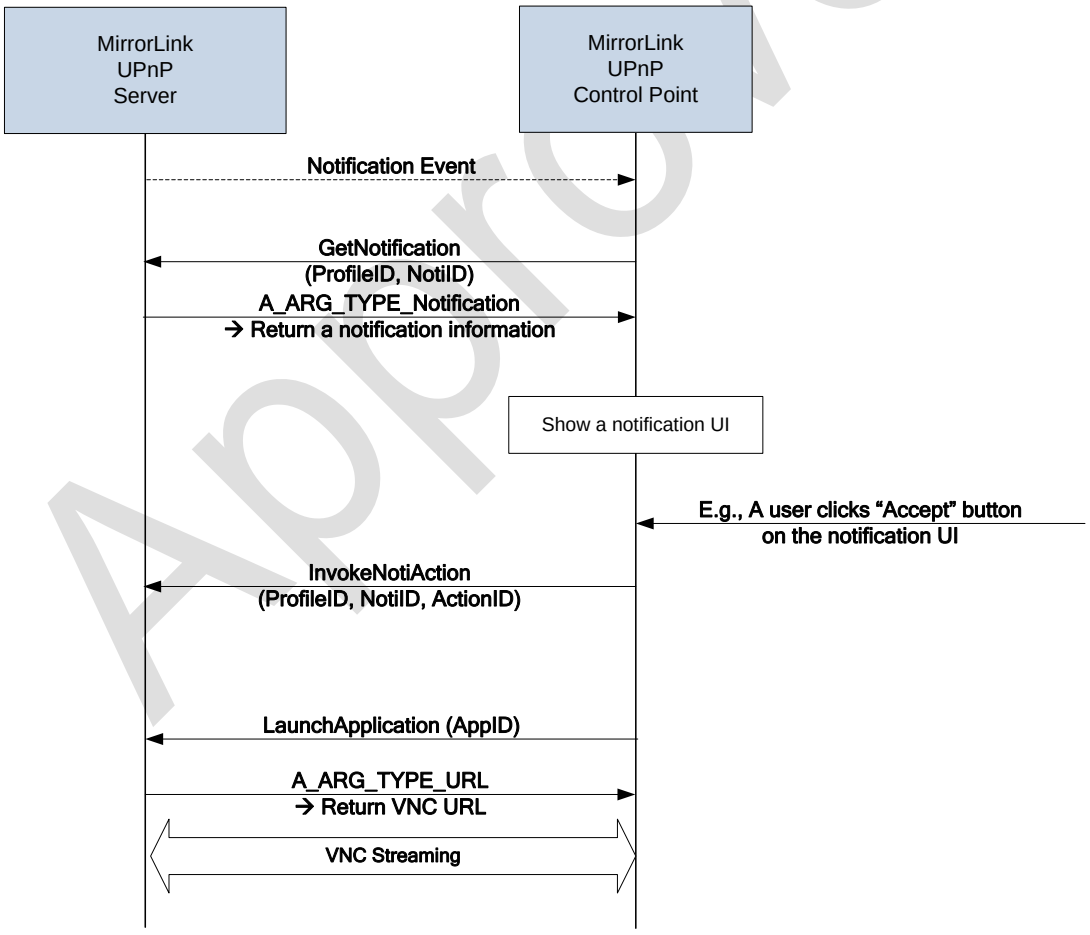

Figure 4: Flow basics to use MirrorLink Client UI for notification

In case the MirrorLink Client does not handle the notification, it MUST use the `InvokeNotiAction` action with the `actionID = 0x00`, in order to clear the notification, as shown in Figure 5.

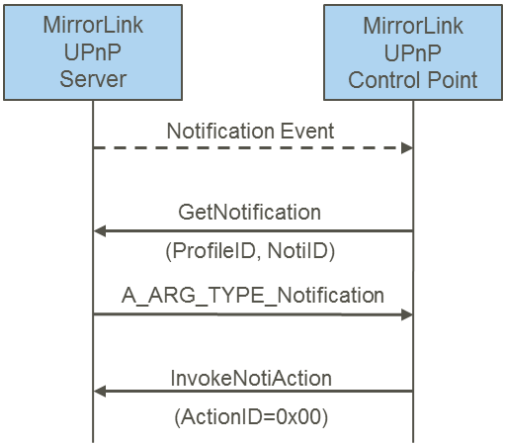

Figure 5: MirrorLink Client clears pending Notification after checking Notification Details

3.3 Displaying a notification message

This section gives an example of a notification for a new text message. The text application provides four actions for the notification such as View, Reply, Delete, and Close in this example.

A_ARG_TYPE_NotiID of the notification (ID: 0x00000002) provided by the text application (ID: 0x00000017) is as follows:

0x00000002@0x00000017

A_ARG_TYPE_Notification of the notification is as follows;

```
<?xml version="1.0" encoding="UTF-8"?>
<notification>
      <notiID>0x00000002@0x00000017</notiID>
      <notiTitle>New Text Message</notiTitle>
      <notiBody>Mark: Where are you at?</notiBody>
      <appID>0x00000017</appID>
      <actionList>
            <action>
                  <actionID>0x00000001</actionID>
                  <actionName>View</actionName>
                  <launchApp>true</launchApp>
            </action>
            <action>
                  <actionID>0x00000002</actionID>
                  <actionName>Reply</actionName>
                  <launchApp>true</launchApp>
            </action>
            <action>
                  <actionID>0x00000003</actionID>
                  <actionName>Delete</actionName>
                  <launchApp>false</launchApp>
            </action>
            <action>
                  <actionID>0x00000004</actionID>
                  <actionName>Close</actionName>
                  <launchApp>false</launchApp>
            </action>
      </actionList>
</notification>
```

The notification specified above will be rendered on the MirrorLink Client as shown in Figure 5.

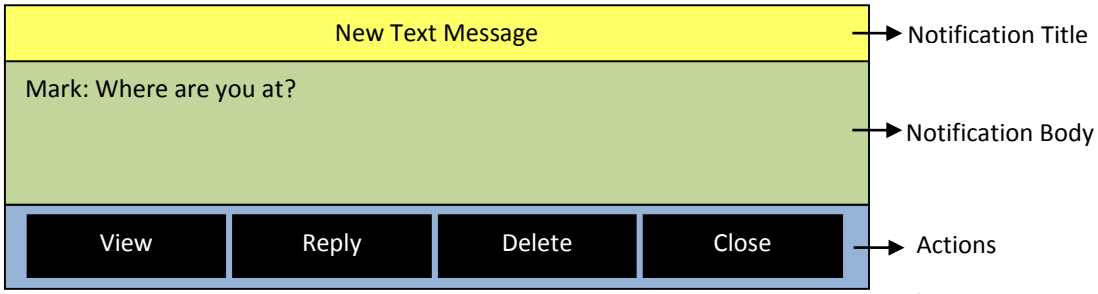

Figure 6: A notification for a new text message event

When a user selects one of the actions provided on the notification UI, the MirrorLink UPnP Control Point MUST invoke InvokeNotiAction() to send ActionID of the selected action to the MirrorLink UPnP Server. Depending on the value of <launchApp> element, the MirrorLink UPnP Control Point decides if invoking LaunchApplication() SOAP action is REQUIRED.

3.4 XML Signature Minimum Set

The MirrorLink Server SHOULD sign the ARG_TYPE_Notification XML.

Signatures MUST follow W3C's recommendation on XML signing, as specified in [5]. 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 `<notification>` element, which is the parent of the `<ds:Signature>` element, for the UPnP Notification 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.

4 A_ARG_TYPE_NOTIFICATION XSD SCHEMA

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="urn:schemas-upnp-
org:tmnotificationserver:notification-1-0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
elementFormDefault="qualified" attributeFormDefault="unqualified"
id="notification">
<xs:import schemaLocation="xmldsig-core-schema.xsd"
namespace="http://www.w3.org/2000/09/xmldsig#"/>
<xs:element name="notification">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="notiID">
    <xs:simpleType>
     <xs:restriction base="xs:string">
      <xs:pattern value="0[Xx][A-Fa-f0-9]{1,8}@0[Xx][A-Fa-f0-9]{1,8}"/>
     </xs:restriction>
    </xs:simpleType>
   </xs:element>
   <xs:element name="notiTitle" type="xs:string"/>
   <xs:element name="notiBody" type="xs:string" default=""/>
   <xs:element name="iconList" minOccurs="0">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="icon" maxOccurs="unbounded">
       <xs:complexType>
        <xs:sequence>
         <xs:element name="mimetype" type="xs:string"/>
         <xs:element name="width" type="xs:positiveInteger"/>
         <xs:element name="height" type="xs:positiveInteger"/>
         <xs:element name="depth" type="xs:positiveInteger"/>
         <xs:element name="url" type="xs:string"/>
         <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: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="actionList" minOccurs="0" maxOccurs="1">
    <xs:complexType>
     <xs:sequence>
```

```
      <xs:element name="action" maxOccurs="unbounded">
       <xs:complexType>
        <xs:sequence>
         <xs:element name="actionID">
          <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="actionName" type="xs:string"/>
         <xs:element name="launchApp" type="xs:boolean"
         default="false"/>
         <xs:element name="iconList" minOccurs="0">
          <xs:complexType>
           <xs:sequence>
            <xs:element name="icon" maxOccurs="unbounded">
             <xs:complexType>
              <xs:sequence>
               <xs:element name="mimetype" type="xs:string"/>
               <xs:element name="width" type="xs:positiveInteger"/>
               <xs:element name="height" type="xs:positiveInteger"/>
               <xs:element name="depth" type="xs:positiveInteger"/>
               <xs:element name="url" type="xs:string"/>
               <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="##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>
```

1

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>GetNotification</name>
         <argumentList>
            <argument>
               <name>ProfileID</name>
               <direction>in</direction>
               <relatedStateVariable>A_ARG_TYPE_ProfileID
               </relatedStateVariable>
            </argument>
            <argument>
               <name>NotiID</name>
               <direction>in</direction>
               <relatedStateVariable>A_ARG_TYPE_NotiID
               </relatedStateVariable>
            </argument>
            <argument>
               <name>Notification</name>
               <direction>out</direction>
               <relatedStateVariable>A_ARG_TYPE_Notification
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>GetSupportedApplications</name>
         <argumentList>
            <argument>
               <name>ProfileID</name>
               <direction>in</direction>
               <relatedStateVariable>A_ARG_TYPE_ProfileID
               </relatedStateVariable>
            </argument>
            <argument>
               <name>AppIDs</name>
               <direction>out</direction>
               <relatedStateVariable>A_ARG_TYPE_String
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>SetAllowedApplications</name>
         <argumentList>
            <argument>
               <name>ProfileID</name>
               <direction>in</direction>
               <relatedStateVariable>A_ARG_TYPE_ProfileID
```

```
                </relatedStateVariable>
             </argument>
             <argument>
                <name>AppIDs</name>
                <direction>in</direction>
                <relatedStateVariable>A_ARG_TYPE_String
                </relatedStateVariable>
             </argument>
          </argumentList>
       </action>
       <action>
          <name>InvokeNotiAction</name>
          <argumentList>
             <argument>
                <name>ProfileID</name>
                <direction>in</direction>
                <relatedStateVariable>A_ARG_TYPE_ProfileID
                </relatedStateVariable>
             </argument>
             <argument>
                <name>NotiID</name>
                <direction>in</direction>
                <relatedStateVariable>A_ARG_TYPE_NotiID
                </relatedStateVariable>
             </argument>
             <argument>
                <name>ActionID</name>
                <direction>in</direction>
                <relatedStateVariable>A_ARG_TYPE_ActionID
                </relatedStateVariable>
             </argument>
          </argumentList>
       </action>
    </actionList>
    <serviceStateTable>
       <stateVariable sendEvents="yes">
          <name>ActiveNotiEvent</name>
          <dataType>string</dataType>
       </stateVariable>
       <stateVariable sendEvents="yes">
          <name>NotiAppListUpdate</name>
          <dataType>string</dataType>
       </stateVariable>
       <stateVariable sendEvents="no">
          <name>A_ARG_TYPE_Notification</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>
       </stateVariable>
       <stateVariable sendEvents="no">
          <name>A_ARG_TYPE_ActionID</name>
          <dataType>string</dataType>
```

```
        </stateVariable>
        <stateVariable sendEvents="no">
           <name>A_ARG_TYPE_NotiID</name>
           <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
           <name>A_ARG_TYPE_String</name>
           <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
           <name>A_ARG_TYPE_URI</name>
           <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
           <name>A_ARG_TYPE_INT</name>
           <dataType>ui4</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
           <name>A_ARG_TYPE_Bool</name>
           <dataType>string</dataType>
        </stateVariable>
     </serviceStateTable>
</scpd>
```

6 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] Car Connectivity Consortium, “MirrorLink – UPnP Application Server Service”, Version 1.1; CCC-TS-024

[4] Car Connectivity Consortium, “MirrorLink – UPnP Client Profile Service”, Version 1.1; CCC-TS-026

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