ETSI TS 103 544-11 V1.3.0 (2017-10)

Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 11: UPnP Notification Server Service

CAUTION

The present document has been submitted to ETSI as a PAS produced by CCC and approved by the ETSI Technical Committee Intelligent Transport Systems (ITS).

Reference
DTS/ITS-88-11

Keywords
interface, ITS, PAS, smartphone

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the only prevailing document is the print of the Portable Document Format (PDF) version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status.
Information on the current status of this and other ETSI documents is available at
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

Copyright Notification

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Intelligent Transport Systems (ITS).

The present document is part 11 of a multi-part deliverable. Full details of the entire series can be found in part 1 [i.1].

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document is part of the MirrorLink® specification which specifies an interface for enabling remote user interaction of a mobile device via another device. The present document is written having a vehicle head-unit to interact with the mobile device in mind, but it will similarly apply for other devices, which provide a colour display, audio input/output and user input mechanisms.

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 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference/.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] UPnP™ Forum: "UPnP™ Device Architecture 1.1", 15 October 2008.

NOTE: Available at http://www.upnp.org.

[2] W3C Recommendation, 10 June 2008: "XML Signature Syntax and Processing (Second Edition)".

NOTE: Available at http://www.w3.org/TR/xmldsig-core/.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI TS 103 544-1 (V1.3.0): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 1: Connectivity".

3 Definitions, symbols and abbreviations

Not applicable.

4 Service Modeling Definition

4.1 Service Type

The following service type identifies a service that is compliant with the present document:

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

TmNotificationServer defined in the present document refers to the same service type. The *TmNotificationServer* service shall follow defined UPnP behaviour within the UPnP Device Architecture 1.1 [1].

4.2 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.

4.3 State Variables

4.3.1 State Variable Overview

Table 4-1: 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

4.3.2 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* shall 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.

> 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 shall 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 Client shall immediately respond to an incoming notification, unless safety related or other higher priority tasks currently require the user's attention. Notifications from unsupported applications shall be immediately cleared, sending an *InvokeNotiAction* with *ActionID* set to `"0x00"`.

Implementation Note:

A MirrorLink 1.1 Client may not immediately respond to an incoming notification.

The MirrorLink UPnP Server shall 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 shall 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 shall 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.

4.3.3 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 shall 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.

4.3.4 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 4-2: 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 shall trim from the right all characters in excess of the specified *notiTitleMaxLength* parameter within the *TmClientProfile* service. (*A_ARG_TYPE_String*)	notification	Required

Element	Description	Parent	Availability
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 characters, like tab, line feed and carriage return. MirrorLink UPnP Control Point shall 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 a 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 shall be either mimetype `image/png` and colour depth 24 or a mimetype and colour depth identical to values set in the client icon preferences as specified using the TmClientProfileServer:1 service's *SetClientProfile* action. • MirrorLink UPnP Control Point shall have support for displaying icons with mimetype `image/png` and colour 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. A MirrorLink Client should launch the application with provided *appID*, without showing any notification, if the *actionList* element is missing.	notification	Optional
action*	Individual action, associated with the notification. MirrorLink UPnP Control Point shall remove from the end all actions in excess of the specified *maxActions* parameter within the *TmClientProfile* service.	actionList	Required
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* shall be sent to the MirrorLink UPnP Server. Shall be non-zero (0x0000) (*A_ARG_TYPE_ActionID*)	action	Required

Element	Description	Parent	Availability
actionName	Action name. This name will be shown as a button label on the native notification UI. MirrorLink UPnP Control Point shall 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 shall be either mimetype `image/png` and colour depth 24 or a mimetype and colour depth identical to values set in the client icon preferences as specified using the TmClientProfileServer:1 service's SetClientProfile action. • MirrorLink UPnP Control Point shall have support for displaying icons with mimetype `image/png` and colour 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 [2]. The key used in calculating the signature shall 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 shall point to notification element. The *SignatureMethod* shall be RSA with SHA1. The *KeyInfo* element may be omitted. The mechanism for generation, exchange and maintenance of keys is out of scope for the present document.	notification	Optional

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

4.3.5 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 shall 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 shall 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)

4.3.6 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.

4.3.7 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 shall 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 shall 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)

4.3.8 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* shall be unique within the given application. If an application runs out of not-used *NotificationIDs*, it shall not send any further notifications.

4.3.9 A_ARG_TYPE_String

A simple string type (UTF-8).

4.3.10 A_ARG_TYPE_URI

A string encoded as UTF-8 representing a URI.

4.3.11 A_ARG_TYPE_INT

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

4.3.12 A_ARG_TYPE_Bool

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

4.4 Eventing and Moderation

Table 4-3 lists the eventing and moderation properties for each of the service state variables.

Table 4-3: 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

4.5 Supporting Multiple Client Profiles

Support for multiple Client Profiles is reserved for future use.

4.6 Actions

4.6.1 General

Table 4-4: Supported actions

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

4.6.2 GetNotification

4.6.2.1 General

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

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

4.6.2.2 Arguments

Table 4-5: 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. Reserved for future. Shall be set to `"0"`.

NotiID (A_ARG_TYPE_NotilD) - Unique notification ID.

Return Value:

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

4.6.2.3 Effect on State

None.

4.6.2.4 Errors

Table 4-6: 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.

4.6.3 GetSupportedApplications

4.6.3.1 General

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

4.6.3.2 Arguments

Table 4-7: 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. Reserved for future. Shall be set 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.

4.6.3.3 Effect on State

None.

4.6.3.4 Errors

Table 4-8: 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.
830	Invalid Profile ID	The profile identifier does not exist or the application cannot use the specified profile identifier.

4.6.4 SetAllowedApplications

4.6.4.1 General

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 *TmApplicationServer:1* service *LaunchApplication* SOAP action or via different means.

The MirrorLink Client, supporting the UPnP Notification Server service, shall support notifications from all CCC and respective Member-certified applications, if they are certified for the respective drive/park mode. The MirrorLink Client should support notifications from base-certified applications in drive mode, if notification handling does not include launching the respective application.

The MirrorLink Client may support notifications from other applications, while being in park mode. The MirrorLink Client shall not support notifications from non-drive certified applications in Drive Mode.

The MirrorLink Client shall set the allowed applications as soon as the MirrorLink Client has retrieved the initial UPnP application listing, or has retrieved an UPnP application listing update.

Implementation Note for MirrorLink 1.1 and 1.2 Clients:

MirrorLink 1.1 and 1.2 Clients may not support all requirements regarding support for notifications from certified applications.

4.6.4.2 Arguments

Table 4-9: 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. Reserved for future. Shall be set 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.

4.6.4.3 Effect on State

None.

4.6.4.4 Errors

Table 4-10: 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.
815	Device Locked	The action cannot be processed as the device hosting the *TmNotificationServer* 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.

4.6.5 InvokeNotiAction

4.6.5.1 General

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

4.6.5.2 Arguments

Table 4-11: 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. Reserved for future. Shall be set 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 shall invoke. If *ActionID* is set to zero (0x0000), the MirrorLink UPnP Server shall not invoke any action. In both cases the active notification is cleared.

Return Value:

None

4.6.5.3 Effect on State

None.

4.6.5.4 Errors

Table 4-12: 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 unlock the device first.
830	Invalid Profile ID	The profile identifier does not exist or the application cannot use the specified profile identifier.

4.6.6 Error Code Summary

Table 4-13 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 4-13: 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.

ErrorCode	errorDescription	Description
810	Bad AppID / Bad NotiID	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*. Bad NotiID • One *NotiID* does not exist or is malformed. • MirrorLink Client should check the *NotiID* (e.g. re-subscribing to the service events) and retry action. • MirrorLink Client should not retry the action with the same *NotiID*.
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 the present document.

5 Theory of Operation

5.1 Initialization steps

The flow chart in Figure 5-1 describes how to initialize the notification service, and how to set the notification filter.

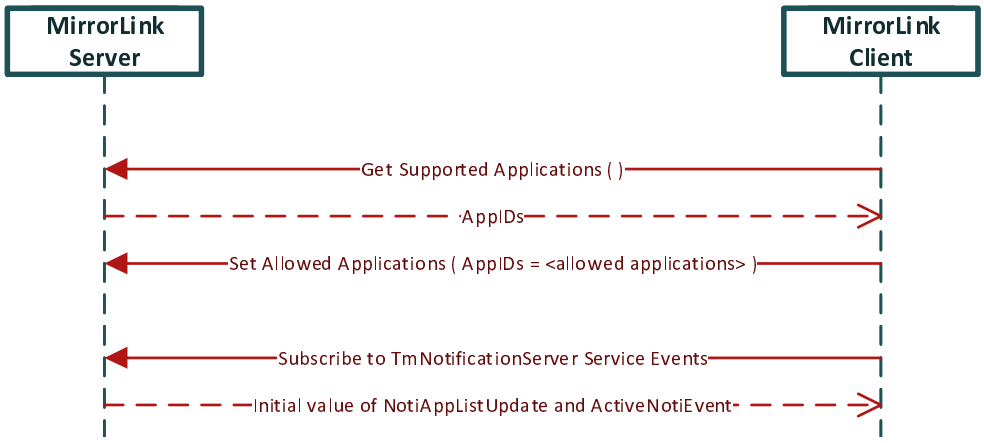

Figure 5-1: Flow basics for initialization

The MirrorLink Server will send a notification either with an action list or without any. Therefore, the MirrorLink Client shall respond in the following way, when receiving a notification:

- Notification without an action list
  - Launch the originating application as specified in clause 5.2.1.
- Notification with an action list
  - Show local notification UI as specified in clause 5.2.2, or
  - Launch the originating application as specified in clause 5.2.1.

A MirrorLink Client, supporting notification, shall support the launch of the originating application. Support for a local application UI is RECOMMENDED.

5.2 Handling of notification

5.2.1 Not using Head Unit UI for notification

The following sequence (see also Figure 5-2) 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.

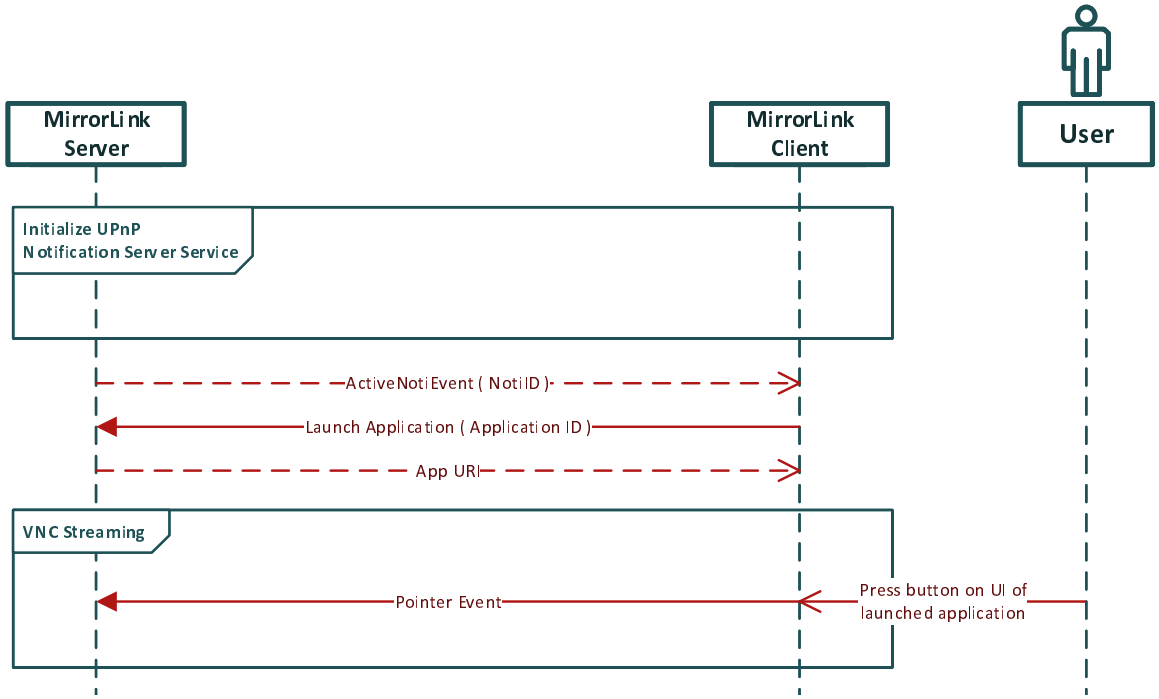

Figure 5-2: Flow basics not to use MirrorLink Client UI for notification

In case the MirrorLink Client does not launch an application, using *LaunchApplication* action, it shall use the *InvokeNotiAction* action with the *actionID* = `0x00`, in order to clear the notification, as shown in Figure 5-3.

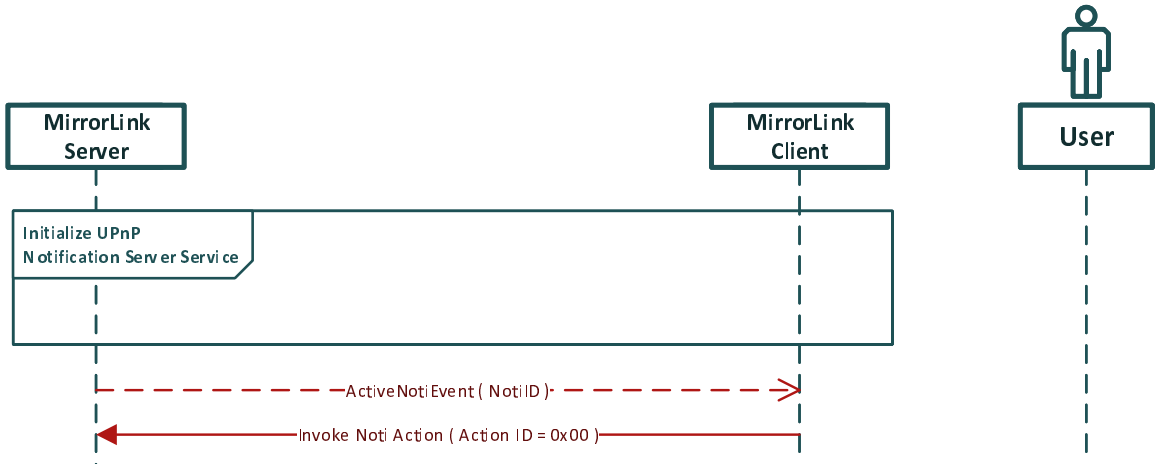

Figure 5-3: MirrorLink Client immediately clears pending Notification

5.2.2 Using Head Unit UI for notification

The following sequence (see also Figure 5-4) 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.

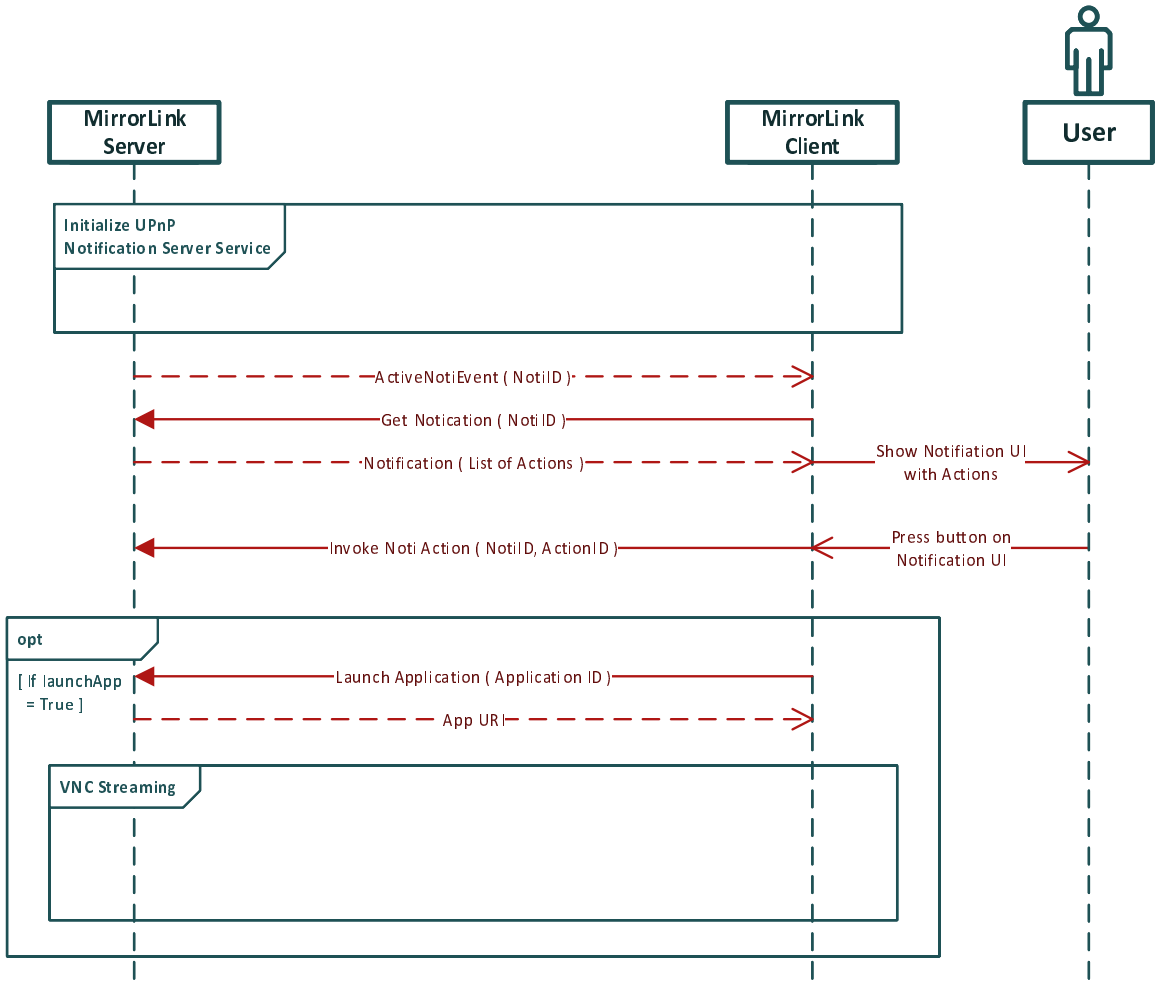

Figure 5-4: Flow basics to use MirrorLink Client UI for notification

The MirrorLink Client will skip steps 3 - 5, if the Notification does not contain an action list and will immediately launch the application with the given *appID* value.

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

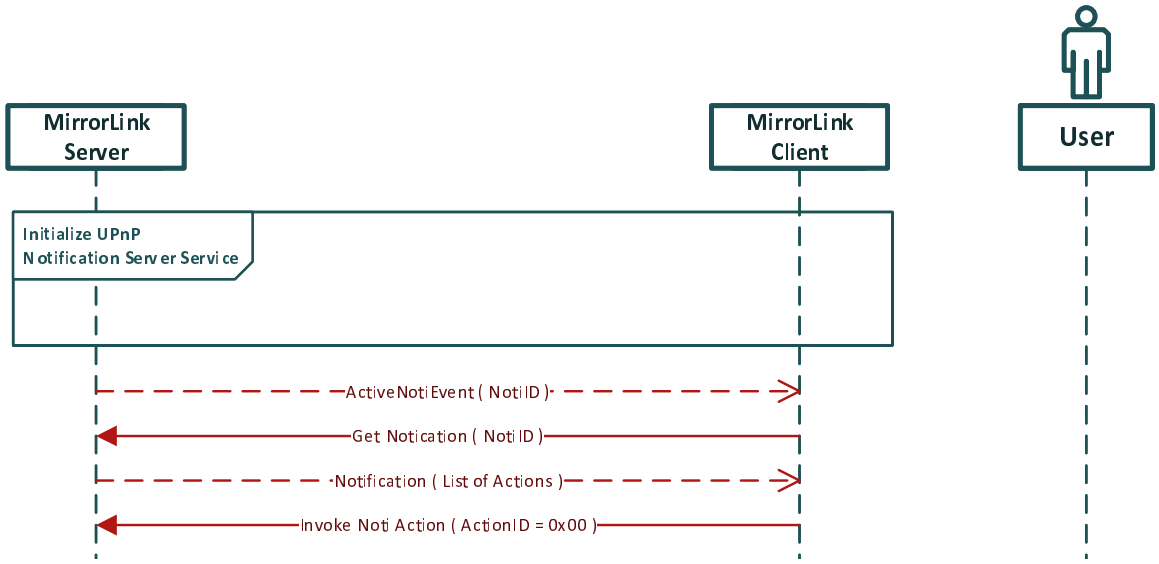

Figure 5-5: MirrorLink Client clears pending Notification after checking Notification Details

5.3 Displaying a notification message

This clause 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 MirrorLink Client, supporting notifications via an embedded native notification UI (*notiUiSupport* is set to `"true"` in the *A_ARG_TYPE_ClientProfile*), it shall support at least the default configuration with respect to *maxActions*, *actionNameMaxLength*, *notiTitleMaxLength* and *notiBodyMaxLength*.

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 5-6.

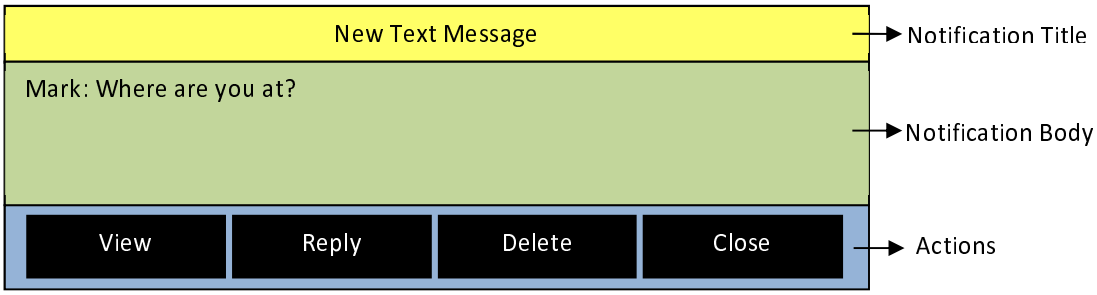

Figure 5-6: A notification for a new text message event

When a user selects one of the actions provided on the notification UI, the MirrorLink UPnP Control Point shall 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.

5.4 XML Signature Minimum Set

The MirrorLink Server should sign the *ARG_TYPE_Notification* XML.

Signatures shall follow W3C's recommendation on XML signing, as specified in [2]. 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 shall be followed for MirrorLink Server and Client devices:

- The **Reference URI** shall not be outside document. It shall 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 shall refer to the element listed above.
- MirrorLink Server shall not use XPath or XSLT **XML transformations.**
- The MirrorLink Server shall use **Canonical method** XML version 1.0 (xml-c14n) or 1.1 (xml-c14n11); Canonical XML version 2.0 or later shall not be used.
- The MirrorLink Server shall use SHA-1 **digest method**; other digest methods shall not be used.
- The MirrorLink Server shall use RSA-SHA1 **signature method**; other signature methods, like HMAC-SHA1 or DSA-SHA1, shall not be used.
- The MirrorLink Client shall not use the **KeyInfo** element to identify a public key to verify the signature; it shall use the *applicationPublicKey* element obtained from the DAP *attestationResponse*, for the `TerminalMode:UPnP-Server` component instead.

6 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:any namespace="##other" 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>
```

7 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>
```

Annex A (informative): Authors and Contributors

The following people have contributed to the present document:

Rapporteur:	Dr. Jörg Brakensiek, E-Qualus (for Car Connectivity Consortium LLC)
Other contributors:	Matthias Benesch, Daimler AG
	Sungjin Lee, Samsung Electronics
	Jungwoo Kim, LG Electronics
	Kyungguen Kim, LG Electronics
	Mingoo Kim, LG Electronics
	Hoyeon Park, Samsung Electronics

History

Document history		
V1.3.0	October 2017	Publication