ETSI TS 103 544-3 V1.3.0 (2017-10)

Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 3: Audio

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-88-3

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 3 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 color display, audio input/output and user input mechanisms.

The present document defines the MirrorLink Audio architecture, based on an RTP forward and back channel, plus an possible Bluetooth HFP and A2DP setup.

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] IETF RFC 3550: "RTP: A Transport Protocol for Real-Time Applications", July 2003, http://tools.ietf.org/html/rfc3550 .

[2] Bluetooth Specification: "Hands-free Profile 1.5", Car Working Group, Revision V10r00, November 25, 2005.

[3] Bluetooth Specification: "Headset Profile", Car Working Group, Revision V12r00, December 18, 2008.

[4] Bluetooth Specification: "Advanced Audio Distribution Profile", Audio Video Working Group.

[5] IETF RFC 2190: "RTP Payload Format for H.263 Video Streams", September 1997, http://tools.ietf.org/html/rfc2190 .

[6] IETF RFC 3551: "RTP Profile for Audio and Video Conferences with Minimal Control", July 2003, http://tools.ietf.org/html/rfc3551 .

[7] ETSI TS 103 544-12 (V1.3.0): "Intelligent Transport Systems (ITS); MirrorLink®; Part 12: UPnP Server Device" .

[8] ETSI TS 103 544-9 (V1.3.0): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 9: UPnP Application Server Service" .

[9] ETSI TS 103 544-10 (V1.3.0): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 10: UPnP Client Profile Service" .

[10] ETSI TS 103 544-15 (V1.3.0): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink® ; Part 15: Application Programming Interface (API) Level 1 & 2".

[11] ETSI TS 103 544-2 (V1.3.0): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink® ; Part 2: Virtual Network Computing (VNC) based Display and Control".

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".

[i.2] ITU-T Recommendation G.114 (05/2003): "One-way transmission time", May 6, 2003; available from http://www.itu.int.

[i.3] ITU-T Recommendation P.1100 (03/2017): "Narrowband hands-free communication in motor vehicles", March 1, 2017; available from http://www.itu.int.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

pointer event: touch screen action in which the user touches the screen with one (virtual) finger only at a single location

touch event: touch screen action in which the user touches the screen with two or more separate fingers at different locations

NOTE: Touch events are used to describe more complex touch action, like pinch-open or pinch-close.

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

A2DP	Bluetooth Advanced Audio Distribution Profile
ARP	Address Resolution Protocol
BT	Bluetooth
CDC	Communications Device Class; specified from USB Device Working Group
CE	Consumer Electronics; CE devices are referred to as mobile devices within the present document
DHCP	Dynamic Host Configuration Protocol
HFP	Bluetooth Hands-free Profile
HSP	Bluetooth Headset Profile
HMI	Human Machine Interface
HU	Head-unit (this term is used interchangeably with the MirrorLink Client)
HS	Head-set
IP	Internet Protocol
LSS	Latency Switching Sources
NCM	Network Control Model; part of the CDC device class
RFB	Remote Framebuffer
RTP	Real-time Transport Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UI	User Interface
UPnP	Universal Plug and Play

USB	Universal Serial Bus
VNC	Virtual Network Computing
WFD	Wi-Fi Display

4 Introduction

The present document defines how the MirrorLink Client selects the transport media that the MirrorLink Server shall use to route audio streams. The MirrorLink Server distinguishes between two audio streams, namely phone and application audio. It advertises available transport options, using UPnP *TmServerDevice*:1 services. Audio streams are specified for the following remote access protocols:

- RTP Real-Time Transport Protocol
- BTA2DP Bluetooth Advanced Audio Distribution Profile
- BTHFP Bluetooth Hands Free Profile

It is the MirrorLink Client's responsibility to select from the advertised audio transport mechanisms how the MirrorLink Server shall stream the different audio sources. The audio link selection is done according the following priorities (highest priority first):

1) Keep existing Bluetooth HFP or A2DP connection to another external device, which is not a MirrorLink Client, if overriding the resource assignment is not allowed.
2) Follow audio link selection using the mechanism described in this clause.
3) Manual Bluetooth pairing (same behaviour as in non-MirrorLink use cases).

MirrorLink Server's speaker shall not be used for audio output.

MirrorLink Server's microphone shall not be used, for voice input if the MirrorLink Client indicates support for voice command within its UPnP Client Profile.

The present document allows for different transport mechanisms based on the selections the MirrorLink Client has taken.

The MirrorLink audio architecture, as shown in Figure 1, allows using the Real-time Transport Protocol for streaming audio captured from the mobile device, to the MirrorLink Client. The audio output from the mobile device is streamed in an application agnostic manner so that it does not require re-design or modification of existing applications running on the device.

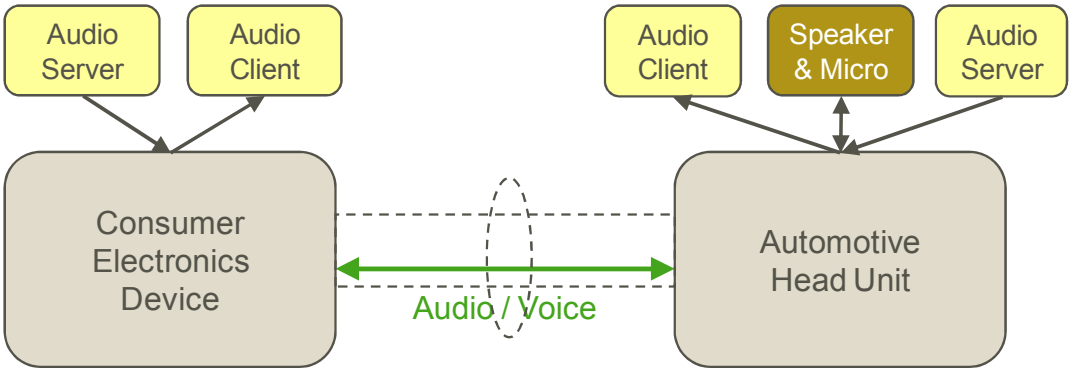


Figure 1: Audio Setup

The present document covers both audio output from and audio input to the MirrorLink Server device. Unless otherwise stated, the present document applies for the audio server and the audio client in the same way:

- The audio output will be handled from the audio server on the MirrorLink Server and the audio client on the MirrorLink Client.

- The audio input will be handled from the audio client on the MirrorLink Server and the audio server on the MirrorLink Client.

In addition to RTP, the MirrorLink specification allows regular Bluetooth audio connectivity for both phone and media audio streams.

5 Audio Link Options

MirrorLink allows the MirrorLink Client and Server to use the following audio features:

- Media – Streaming of media audio from the MirrorLink Server to the MirrorLink Client; media audio feature includes navigation audio.
- Phone – Bidirectional streaming of conversational audio, including in-band ringing.
- VC – Streaming of voice command (VC) audio from the MirrorLink Client to the MirrorLink Server.

The above listed audio features are available over the following connectivity options:

- Bluetooth (BT)
- RTP

The MirrorLink Server shall advertise the audio features and connectivity options it supports within the UPnP *A_ARG_TYPE_AppList* application listings. The MirrorLink Server shall advertise individual audio components, with all combinations of the supported audio content categories.

EXAMPLE: If the MirrorLink supports an RTP Server for Media and Phone Audio, it shall advertise three RTP Servers, one showing Media, one Phone, and one Media & Phone support.

This will allow the MirrorLink Client to specifically inform the MirrorLink Server, which feature it is going to use. The possible audio content categories for the different audio components are listed below:

- RTP Client: Phone Audio, Voice Command In, Phone Audio + Voice Command In
- RTP Server: Media Audio Out, Phone Audio + Media Audio Out
- BT A2DP: Media Audio Out
- BT HFP: Phone Audio, Phone Audio + Voice Command In

The MirrorLink Server shall include the RTP Server with a value of *audioInfo@contentCategory* equaling `Media Audio Out (0x01)` first in the list of advertised RTP Servers for the same RTP payload types (*A_ARG_TYPE_AppList*).

If the MirrorLink Server supports Voice Command over RTP, it shall include the RTP Client with a value of *audioInfo@contentCategory* equaling `Voice Command In (0x10)` first in the list of advertised RTP Clients for the same RTP payload types (*A_ARG_TYPE_AppList*).

The MirrorLink Client shall select the audio features and their connectivity option, which the MirrorLink Client is going to use. The MirrorLink Client shall use the UPnP Application Server service's Launch Application action to inform the MirrorLink Sever about its selection. The MirrorLink Client shall not launch more than one RTP Client, RTP Server, BT HFP, and BT A2DP component.

Based on the set of audio components, launched from the MirrorLink Client, the MirrorLink Server and Client shall use specific audio links for Media, Phone and Voice Command use cases. Table 1 lists the possible combinations of audio features and their underlying connectivity options on the left side. For each combination, the required audio components are listed, which shall be launched from the MirrorLink Client to enable it. Note, that some Audio Feature Set combinations are not possible. The below table shall not be used in case a WFD session, introduced in MirrorLink 1.2, is established.

Table 1: UPnP Negotiation for Audio Selection (Non-WFD Case)

Audio Feature Combinations and underlying Connectivity			Audio Components launched from the MirrorLink Client to enable the Audio Feature Combination			
Media	**Phone**	**VC**	**BT HFP**	**BT A2DP**	**RTP Server**	**RTP Client**
-	-	-	-	-	-	-
-	-	RTP	-	-	-	VC
-	-	BT	Not possible			
-	RTP	-	-	-	Phone	Phone
-	RTP	RTP	-	-	Phone	Phone + VC
-	RTP	BT	Not possible			
-	BT	-	Phone	-	-	-
-	BT	RTP	Phone	-	-	VC
-	BT	BT	Phone + VC	-	-	-
RTP	-	-	-	-	Media	-
RTP	-	RTP	-	-	Media	VC
RTP	-	BT	Not possible			
RTP	RTP	-	-	-	Phone + Media	Phone
RTP	RTP	RTP	-	-	Phone + Media	Phone + VC
RTP	RTP	BT	Not possible			
RTP	BT	-	Phone	-	Media	-
RTP	BT	RTP	Phone	-	Media	VC
RTP	BT	BT	Phone + VC	-	Media	-
BT	-	-	-	Media	-	-
BT	-	RTP	-	Media	-	VC
BT	-	BT	Not possible			
BT	RTP	-	-	Media	Phone	Phone
BT	RTP	RTP	-	Media	Phone	Phone + VC
BT	RTP	BT	Not possible			
BT	BT	-	Phone	Media	-	-
BT	BT	RTP	Phone	Media	-	VC
BT	BT	BT	Phone + VC	Media	-	-

NOTE: Entries marked as `Not possible` in these tables are meant to describe combinations, which are out of scope from the MirrorLink specification. Behaviour, if supported, is implementation dependent. Entries should not be used.

BT HFP and BT A2DP may be connected outside the MirrorLink session, i.e. without specifically using the UPnP mechanisms to launch those. In that case the MirrorLink Client and Server shall make the selection based on the following table. Components marked with `Legacy` shall not be launched using the UPnP *TmApplicationServer* service's *LaunchApplication* action. The setup of the legacy Bluetooth connection is outside the scope of the MirrorLink specifications. The below table shall not be used in case a WFD session, introduced in MirrorLink 1.2, is established.

Table 2: UPnP Negotiation for Audio Selection with Legacy Bluetooth (Non-WFD Case)

Audio Feature Set and underlying Connectivity			Audio Components launched from the MirrorLink Client to enable the Audio Feature Set			
Media	**Phone**	**VC**	**BT HFP**	**BT A2DP**	**RTP Server**	**RTP Client**
-	BT	-	Legacy	-	-	-
-	BT	RTP	Legacy	-	-	VC
-	BT	BT	Legacy	-	-	-
RTP	BT	-	Legacy	-	Media	-

Audio Feature Set and underlying Connectivity			Audio Components launched from the MirrorLink Client to enable the Audio Feature Set			
Media	**Phone**	**VC**	**BT HFP**	**BT A2DP**	**RTP Server**	**RTP Client**
RTP	BT	RTP	Legacy	-	Media	VC
RTP	BT	BT	Legacy	-	Media	-
BT	-	-	-	Legacy	-	-
BT	-	RTP	-	Legacy	-	VC
BT	-	BT	Not possible			
BT	RTP	-	-	Legacy	Phone	Phone
BT	RTP	RTP	-	Legacy	Phone	Phone + VC
BT	RTP	BT	Not possible			
BT	BT	-	Legacy	Legacy	-	-
BT	BT	RTP	Legacy	Legacy	-	VC
BT	BT	BT	Legacy	Legacy	-	-

If an audio use feature is not covered from the audio component selection, done by the MirrorLink Client, the MirrorLink Server shall fallback to its default configuration.

MirrorLink 1.2 introduces Wi-Fi Display (WFD) based audio/video streaming for MirrorLink. The use of WFD removes the need to setup a separate RTP forward channel to carry media audio. The MirrorLink Server shall stream media audio via WFD RTP streaming.

Based on the set of audio components, launched from the MirrorLink Client, the MirrorLink Server and Client shall use specific audio links for additional Phone and Voice Command use cases. Table 1 lists the possible combinations of audio features and their underlying connectivity options on the left side. For each combination, the required audio components are listed, which shall be launched from the MirrorLink Client to enable it. Note, that some Audio Feature Set combinations are not possible.

Table 3: UPnP Negotiation for Audio Selection (WFD Case)

Audio Feature Combinations and underlying Connectivity			Audio Components launched from the MirrorLink Client to enable the Audio Feature Combination			
Media	**Phone**	**VC**	**BT HFP**	**BT A2DP**	**RTP Server**	**RTP Client**
-	{ -, WFD, BT }	{ -, RTP, BT }	Not possible			
WFD	-	-	-	-	{ -, Media }	-
WFD	-	RTP	-	-	{ -, Media }	VC
WFD	-	BT	Not possible			
WFD	WFD	-	-	-	{ -, Media }	Phone
WFD	WFD	RTP	-	-	{ -, Media }	Phone + VC
WFD	WFD	BT	Not possible			
WFD	BT	-	Phone	-	{ -, Media }	-
WFD	BT	RTP	Phone	-	{ -, Media }	VC
WFD	BT	BT	Phone + VC	-	{ -, Media }	-
BT	{ -, WFD, BT }	{ -, RTP, BT }	Not possible			

The MirrorLink Server shall not stream media audio content via a separate RTP Server, even if that RTP Server has been individually launched from the MirrorLink Client. The MirrorLink Client may launch the MirrorLink Server's RTP Server to allow for a faster handover from a WFD to a Non-WFD based audio streaming.

The MirrorLink Server shall transition media audio content to the RTP Server, if it has been launched from the MirrorLink Client, when the WFD session is disconnected. The MirrorLink Server shall transition media audio content from a launched RTP Server (or BT A2DP) to WFD, once the WFD session is established.

BT HFP may be connected outside the MirrorLink session, i.e. without specifically using the UPnP mechanisms to launch those. In that case the MirrorLink Client and Server shall make the selection based on the following table. Components marked with `Legacy` shall not be launched using the UPnP Application Server service's Launch Application action. The setup of the legacy Bluetooth connection is outside the scope of the MirrorLink specifications.

Table 4: UPnP Negotiation for Audio Selection with Legacy Bluetooth (WFD Case)

Audio Feature Set and underlying Connectivity			Audio Components launched from the MirrorLink Client to enable the Audio Feature Set			
Media	**Phone**	**VC**	**BT HFP**	**BT A2DP**	**RTP Server**	**RTP Client**
-	{ -, WFD, BT }	{ -, RTP, BT }	Not possible			
WFD	BT	-	Legacy	-	{ -, Media }	-
WFD	BT	RTP	Legacy	-	{ -, Media }	VC
WFD	BT	BT	Legacy	-	{ -, Media }	-
BT	{ -, WFD, BT }	{ -, RTP, BT }	Not possible			

BT A2DP shall not be used in a WFD session. A legacy BT A2DP connection shall be disconnected.

6 Audio Link Selection

6.1 General

The *TmApplicationServer*:1's service shall be used for controlling audio streams. Each type of audio source or sink on the MirrorLink Server is considered a remote application which can be remotely controlled by the MirrorLink Control Point using TmApplicationServer:1 service.

The audio servers and clients can be started and terminated the same way as any other remote application using the *LaunchApplication*() and *TerminateApplication*() SOAP actions respectively. The audio server, optionally running as part of the MirrorLink Client, will provide audio input like voice control to the MirrorLink Server.

Next a description is provided of how *TmApplicationServer*:1's SOAP actions are utilized to select the audio links. Note that only the aspects specific to audio link selection are covered here and the reader is REQUIRED to refer to the corresponding service specification for complete details of the TmApplicationServer:1.

The MirrorLink Client should make the audio link selection not later than 10 s after receiving the first *A_ARG_TYPE_AppList* response from the MirrorLink Server.

The MirrorLink Client shall have made the audio link selection prior starting the first VNC based remote application.

6.2 Identification of Available Audio Links

The identification of audio links is described in [8].

6.3 LaunchApplication (AppID, ProfileID)

The *LaunchApplication*() action shall be used to start the audio streaming on the MirrorLink Server side and therefore select the underlying audio link. The response received will be an URI to the audio streaming sources/sinks using the audio streaming protocol identifier. The URI shall follow the A_ARG_TYPE_URI definition as specified in [8].

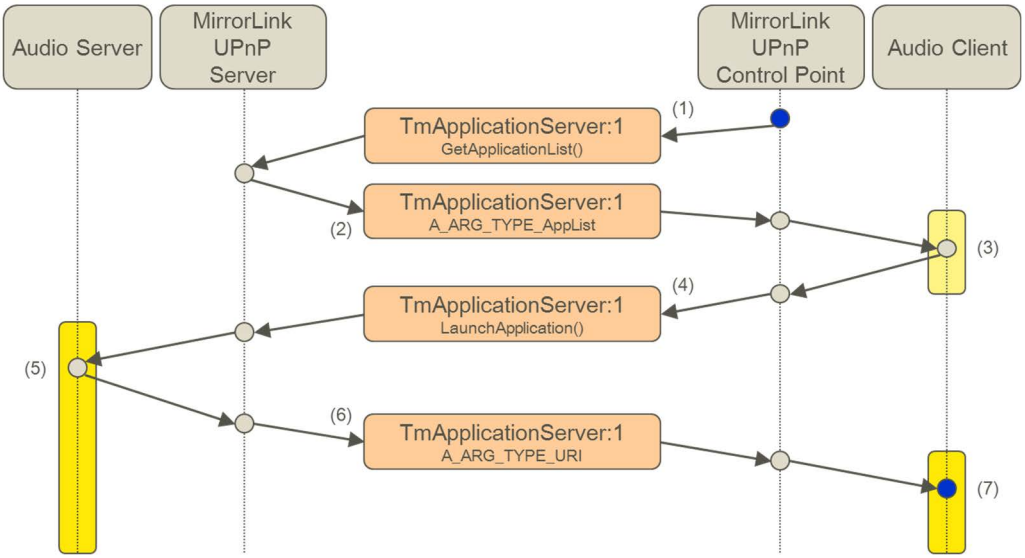


Figure 2: Message Flow – Launch Audio Link

The message flow for selecting an audio link, using *LaunchApplication*(), as shown in Figure 2, consists of the following steps:

1) MirrorLink UPnP Control Point: Call *GetApplicationList*() SOAP action.

2) MirrorLink UPnP Server: Return *A_ARG_TYPE_AppList*, which includes a list of available audio servers.

3) MirrorLink Client: Select the preferred audio server, using the *protocolID* and direction elements:

   a) BT only: Prepare for BT connection, if MirrorLink Server is expected to initiate the BT connection.

NOTE: The MirrorLink Client can indicate to the server via the *SetClientProfile* action that the server has to initiate the BT connection. By default, the MirrorLink Server will not start the BT connection.

4) MirrorLink UPnP Control Point: Call *LaunchApplication*() SOAP action containing respective audio server application ID

5) MirrorLink Server: Start selected audio server:

   a) Determine new audio link configuration, using Table 1.

   b) RTP only: Prepare RTP streaming; RTP streaming is done over UDP.

   c) BT only: Prepare for BT connection.

   d) BT only: Initiate BT connection if MirrorLink Server is expected to initiate the BT connection.

6) MirrorLink UPnP Server: Return A_ARG_TYPE_URI representing the audio server URI

7) MirrorLink Client: Start audio client:

   a) RTP only: Connect to RTP streaming port.

   b) BT only: Initiate BT connection, if MirrorLink Client is expected to initiate the BT connection.

Using *LaunchApplication*() will enable a specific audio link. This link is valid until *TerminateApplication*() action with the same *AppID* is called, or the MirrorLink session is closed.

If Bluetooth is already turned on, the MirrorLink Server shall accept BT connection requests after responding to launching a specific Bluetooth profile. If Bluetooth is turned off, the MirrorLink Server may switch on Bluetooth before responding to launching a specific Bluetooth profile.

The MirrorLink Server shall only initiate a Bluetooth connection, if the MirrorLink Client has specifically requested the server to do so, setting <*startConnection*> in *A_ARG_TYPE_ClientProfile* to `"false"`. To ensure automatic pairing, without user intervention, the MirrorLink Client should provide its Bluetooth MAC address (*bdAddr*) in *A_ARG_TYPE_ClientProfile*.

If the MirrorLink Server does not support Bluetooth connection initialization, i.e. it has specifically set <*startConnection*> in the UPnP TmServerDevice:1 device XML to `"false"` [7], and the MirrorLink Client does not support Bluetooth connection initialization either, the MirrorLink Client shall not use the *LaunchApplication*() to start a Bluetooth connection.

6.4 TerminateApplication (AppID, ProfileID)

The *TerminateApplication*() action shall be used to stop the audio streaming (in either direction) on the MirrorLink Server side and follows the specification [8]. Invoking the *TerminateApplication* action causes the corresponding audio link to be closed.

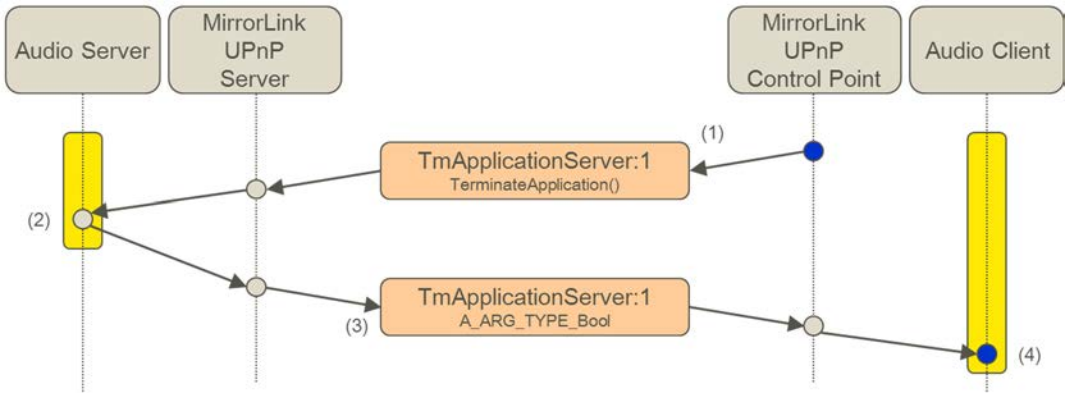


Figure 3: Message Flow – Terminate Audio Link

The message flow for unselecting an audio link, using *TerminateApplication*(), as shown in Figure 3, consists of the following steps:

1) MirrorLink UPnP Control Point: Call *TerminateApplication*() SOAP action containing respective audio server application ID
2) MirrorLink Server: Stop audio server
   a) Determine new audio link configuration, using Table 1
   b) RTP only: Stop the RTP streaming
   c) BT only: Disconnect BT profile; optionally power down BT
3) MirrorLink UPnP Server: Return termination success response (true or false)
4) MirrorLink Client: Stop audio client
   a) RTP only: Disconnect from RTP streaming port.
   b) BT only: Disconnect BT profile; optionally power down BT.

In case the *TerminateApplication*() action is used to terminate the audio server residing in the MirrorLink Server device then the MirrorLink Client will stop receiving the audio stream from the MirrorLink Server.

In case the *TerminateApplication*() action is used to terminate the audio client residing in the MirrorLink Server then the MirrorLink Server will stop receiving the audio stream from the MirrorLink Client.

6.5 GetApplicationStatus (AppID, ProfileID)

The *GetApplicationStatus*() action provides the current status of an audio server or client running on the MirrorLink Server and is following [8].

The return values (of the type *A_ARG_TYPE_AppStatus*) specified for this SOAP action can be any of the following:

- `Foreground`: Audio link is launched.
- `Background`: Not used.
- `Notrunning`: Audio link is terminated / not launched.

7 RTP Audio Streaming

7.1 General

The audio RTP server and client on the MirrorLink Server listen on pre-specified ports, which are advertised using UPnP mechanisms.

When the audio server captures audio data, it will encode the audio into RTP packets using the negotiated RTP Payload type and transmit the RTP packets over UDP/IP.

The audio client is fully responsible for receiving and decoding the data packets and restoring the packet order if they arrive in out of order sequence. More detailed information about the RTP packet structure can be found later in the present document.

The MirrorLink Server shall support RTP audio streaming for unidirectional audio to the MirrorLink Client. The MirrorLink Client shall support RTP audio streaming for unidirectional audio from the MirrorLink Server. The MirrorLink Server may support RTP audio streaming for bi-directional audio.

NOTE: If bi-directional RTP streaming is not supported, telephony use cases work only if BT HFP is supported.

7.2 RTP Packet Structure and Header Definition

RTP packets contain the standard RTP message header and the payload. Usually each RTP packet audio payload contains a predefined amount of audio data, but in a special case of end of stream (M=1), payload can be zero length. Therefore, the RTP client should not assume fixed payload length. Each RTP packet audio payload contains predefined amounts of audio data. Audio samples are sent in sequential order (in sampling order, first sample first).

Each RTP packet contains the standard header as defined in IETF RFC 3550 [1]. The header fields and their default values are described in the following clause. The RTP packet structure is shown in Table 5.

Table 5: RTP Packet Header Definition

00	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
V		P	X	CC				M	PT							Sequence Number															
Timestamp																															
SSRC																															
CSRC [0..15]																															

NOTE: The RTP header definition keeps the IETF format for numbering bits. Therefore, the most significant bit is 0 for all RTP header and payload descriptions.

The first twelve octets are present in every RTP packet, while the list of CSRC identifiers is present only when inserted by a mixer. The following fields are the RTP specification, unless otherwise mentioned:

- **Version (V): 2 bits**

  The RTP version defined by the present document is two (2).

- **Padding (P): 1 bit**

  If the padding bit is set, the packet contains one or more additional padding octets at the end which are not part of the payload.

- **Extension (X): 1 bit**

  If the extension bit is set, the fixed header shall be followed by exactly one header extension. If the RTP header carries information about the audio category and application id, then this bit shall be one (1).

- **CSRC Count (CC): 4 bits**

  The CSRC count contains the number of CSRC identifiers that follow the fixed header.

- **Marker (M): 1 bit**

  The interpretation of the marker is defined (in reference to IETF RFC 2190 [5]);

  0: More packets will follow.

  1: Current package carries the end of stream; audio client may use this information to e.g. empty any available receive buffer.

- **Payload Type (PT): 7 bits**

  This field identifies the format of the RTP payload and determines its interpretation by the application.

- **Sequence Number: 16 bits**

  The sequence number increments by one for each RTP data packet sent, and may be used by the receiver to detect packet loss and to restore packet sequence.

- **Timestamp: 32 bits**

  The timestamp reflects the sampling instant of the first octet in the RTP data packet. The initial value of the timestamp is random.

- **SSRC Synchronization Source: 32 bits**

  This field identifies the synchronization source.

- **CSRC Contributing Source: 32 bits**

  An array of 0 to 15 CSRC elements identifying the contributing sources for the payload contained in this packet. The number of identifiers is given by the *CC* field. should not be used, as the RTP extension header contains respective information.

If the RTP header carries information about the audio sources, the RTP extension header shall be used as shown in Table 6. The RTP header extension shall follow clause 5.3.1 of IETF RFC 3550 [1].

Table 6: RTP Packet Header Extension

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Profile Identifier																Length															
Header Extension::Application Identifier																															
Header Extension::Application Category																															

- **Profile Identifier: 16 bits**

  Profile identifier for the extension header; shall be 0x388C.

- **Length: 16 bits**

  Number of 32-bit words for the extension, excluding the extension header (therefore 0 is a valid length). Length shall be an even number.

- **Header Extension: Array of 64 bits**

  In case of a length of greater than zero, the Header Extension consists of one or more entries of 64 bits each. Each 64-bit entry references one application contributing to the audio content within the RTP payload. The entry with the highest priority shall be given first. Other entries shall follow in decreasing priority order. Priority order is defined by the weights of the audio entries within the audio mixing stage.

  The MirrorLink Server should only list significant audio sources:

  - The first 32 bits give the unique application id. For an application being advertised via UPnP, the unique application id shall match the advertised *appID*. This field may be left empty (i.e. zero value).

  - The second 32 bits give the application category, as defined in [8].

The audio server shall use RTP Packet Header Extensions to provide the application category and the application identifier of the transmitted audio stream. The RTP stream shall contain only a single audio stream. In case multiple audio sources contribute to the stream, the audio server shall mix them together prior streaming to the audio sink; i.e. the RTP stream shall not contain multiplexed, individual audio streams.

The MirrorLink Server shall use the audio context information, provided from MirrorLink certified and MirrorLink-aware applications via the MirrorLink API [10]. The MirrorLink Server shall use the correct application identifier of the application, using the MirrorLink API.

Otherwise for applications not using the MirrorLink API, the MirrorLink Server should determine the source of the application audio streams or shall set the application identifier to 0x000000000, and shall set the audio context information to 0x00000000.

RTP packets without a valid RTP Header Extension should be treated like RTP packets received from an unknown application.

The present document defines the following audio categories, which are used from MirrorLink aware applications, representing different kind of audio.

Table 7: Audio Categories

Audio Category	Audio Category Use Case
`0x00030000` - General Media	• Default for audio, not otherwise specified below • Used for unknown audio category ids • Sound for entertaining the driver • Music, Podcasts, Streaming, . . . • Normally without a pre-defined end
`0x00050000` - Navigation	• Short sound bites for navigation related information
`0x00080000` - Announcements	• Short sound bites for informing the driver
`0xF0000020` - Telephone	• Undefined length • Especially for audio
`0xF0000010` - Speech	• Short audio interactions between server and client

The MirrorLink Client shall follow the priority order, given in Table 8, unless a higher priority event happens on the MirrorLink Client.

Implementation Note

MirrorLink 1.0 and older MirrorLink 1.1 Clients may implement a different priority scheme. In case a MirrorLink application is using a different audio category, the MirrorLink Client should treat it like General Media (Entertainment) audio. In those cases, the consumer will need to manually switch audio sources.

Table 8 shows, in a matrix form, whether the audio from the MirrorLink Server (marked as Server) or the MirrorLink Client (marked as Client) shall be played via the MirrorLink Client's speaker, dependent of the audio categories. For some combinations, this is implementation dependent (marked as Vendor specific). The priority of the Audio content is increasing from the left to the right (from MirrorLink Server perspective and from top to bottom (from MirrorLink Client perspective).

The `"Unknown"` column identifies an audio stream for which the MirrorLink Server (or the MirrorLink Application) has not provided sufficient audio context information. In addition, a MirrorLink Server may treat audio from applications, not having the right certification level, like `"Unknown"` audio, even if audio context information is provided.

Table 8: Audio Channel Priority Table (as implemented by the MirrorLink Client)

		MirrorLink Server						
		No Audio	**Unknown**	**Media**	**Announcement**	**Speech**	**Navigation**	**Telephone**
MirrorLink Client	**No Audio**	N/A	Vendor specific	Server	Server	Server	Server	Server
	Media	Client	Vendor specific	Last one wins	Server	Server	Server	Server
	Announcement	Client	Client	Client	Vendor specific	Server	Server	Server
	Speech	Client	Client	Client	Vendor specific	Vendor specific	Server	Server
	Navigation	Client	Client	Client	Client	Client	Vendor specific	Server
	Telephone	Client	Client	Client	Vendor specific	Vendor specific	Vendor specific	Vendor specific
	Safety	Client	Client	Client	Client	Client	Client	Client

Implementation Note

MirrorLink 1.0 Servers may exclude audio context information from the RTP extension header (`"Unknown Audio"`). MirrorLink Clients supporting MirrorLink 1.0 Servers should support an incoming MirrorLink Audio stream as well, including an audio stream marked as `"Unknown Audio"` (Most MirrorLink 1.0 Servers, incorrectly fail to provide adequate audio context information in the RTP extension header). This should include giving the incoming audio stream higher priority than local entertainment, in case a navigation application has been launched on the MirrorLink Server and that application is in the foreground.

7.3 RTP Audio Payload Definition

7.3.1 General

RTP payload length and sampling frequency shall be negotiated beforehand. This shall be done using UPnP mechanisms as defined in [8] and [9]. The following paragraphs define the audio payload format for 8 and 16 bit audio samples.

The RTP server should support payload type 0 (8 bit, 8 kHz, mono). Audio stream is mu-law encoded [6]. The RTP server shall support payload type 99 (16 bit, stereo, 48 kHz). The RTP client shall support payload type 99. The audio server and client may support other standardized RTP payload types as well. The RTP server shall use a payload type, supported from the RTP client.

A MirrorLink Server, supporting Voice Commands, shall support payload type 100 for its RTP Client. A MirrorLink Client, supporting Voice Commands, shall support payload type 100 for its RTP Server.

7.3.2 kHz, 16 Bit Audio Payload (Mono)

This payload type denotes uncompressed audio data samples using 16-bit signed representation with 65 535 equally divided steps between minimum and maximum signal levels, ranging from -32 768 to 32 767. The value is represented in two's complement notation and transmitted in network byte order (most significant byte first).

The audio data has the following properties:

- One audio channels (mono)
- 16 bits
- Frequency: 48 kHz
- Payload type: 98

Each audio sample is stored to the RTP payload using 16 bits. Each sample is stored to the payload using the order it was taken (first sample first).

Table 9: RTP Payload Format – 16 Bit Mono

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
Sample #(n-1)															
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
Sample #n															

NOTE: Except for the sampling frequency, this payload type is identical to payload type 11 (L16, mono, 44,1 kHz), as defined in IETF RFC 3551 [6].

7.3.3 kHz, 16 Bit Audio Payload (Stereo)

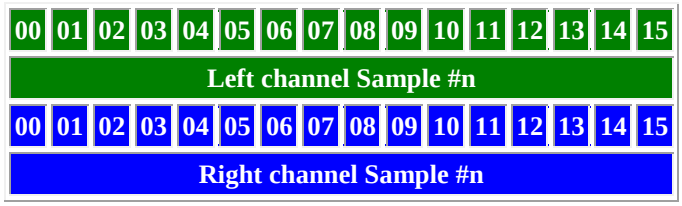
This payload type denotes uncompressed audio data samples using 16-bit signed representation with 65 535 equally divided steps between minimum and maximum signal level, ranging from -32 768 to 32 767. The value is represented in two's complement notation and transmitted in network byte order (most significant byte first).

The audio data has the following properties:

- Two audio channels (stereo).
- 32 bits (16 bits per channel).
- Frequency: 48 kHz
- Audio data for each channel interleaved.
- Payload type: 99

Each audio sample is stored to the RTP payload using 32 bits. Each sample is stored to the payload using the order it was taken (first sample first). Audio sample's left channel data (16 bits) is stored first and then the right channel data (16 bits). This process is applied for each of the audio samples. Audio payload always contains both the right and left channel data, as shown in Table 10. Channel data is never divided amongst different RTP packets.

Table 10: RTP Payload Format – 16 Bit Stereo



NOTE: Except for the sampling frequency, this payload type is identical to payload type 10 (L16, stereo, 44,1 kHz), as defined in IETF RFC 3551 [6].

7.3.4 kHz, 16 Bit Audio Payload (Mono)

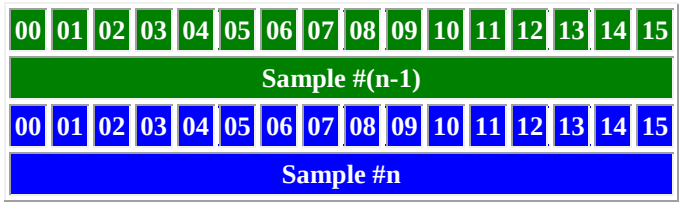
This payload type denotes uncompressed audio data samples using 16-bit signed representation with 65 535 equally divided steps between minimum and maximum signal levels, ranging from -32 768 to 32 767. The value is represented in two's complement notation and transmitted in network byte order (most significant byte first).

The audio data has the following properties:

- One audio channel (mono)
- 16 bits
- Frequency: 16 kHz
- Payload type: 100

Each audio sample is stored to the RTP payload using 16 bits. Each sample is stored to the payload using the order it was taken (first sample first).

Table 11: RTP Payload Format – 16 Bit Mono



NOTE: Except for the sampling frequency, this payload type is identical to payload type 11 (L16, mono, 44,1 kHz), as defined in IETF RFC 3551 [6].

7.4 Establishing the RTP Connection

7.4.1 General

RTP streaming in MirrorLink is based on UDP, which is connectionless. Some level of initial connection is REQUIRED for the MirrorLink RTP Server to know the MirrorLink RTP Client's IP address and port number.

7.4.2 RTP Server within MirrorLink Server

When the MirrorLink Server takes on the role of the MirrorLink RTP server, the MirrorLink Client shall send an UDP packet, containing 1 byte to the port number and IP address assigned for the MirrorLink Server's audio server.

NOTE: The content of the byte can be any arbitrary value.

On reception of that UDP packet, the audio server shall determine the IP address and port number of the audio client.

After receiving that packet, the audio server within the MirrorLink Server's RTP Server will start RTP streaming to the RTP Client using the port number, for the received 1byte packet, when audio stream is available.

The message flow, as shown in Figure 4, consists of the following steps:

1) RTP server: Wait for a UDP packet with server's start.

2) RTP client: Send 1 byte UDP packet to the RTP server; the RTP server address is obtained through the *TmApplicationServer* UPnP service.

3) RTP client: Get ready for receiving RTP packets.

4) RTP server: Determine client's IP address and port number.

5) RTP server: Begin RTP streaming; if no audio data is available, the RTP server should send a single RTP packet, without data payload

6) RTP client receives RTP packets.

7) RTP client: If the RTP client does not receive any RTP packets it shall continue sending 1 byte UDP packets to the RTP server (as defined in step 2) at regular time intervals.

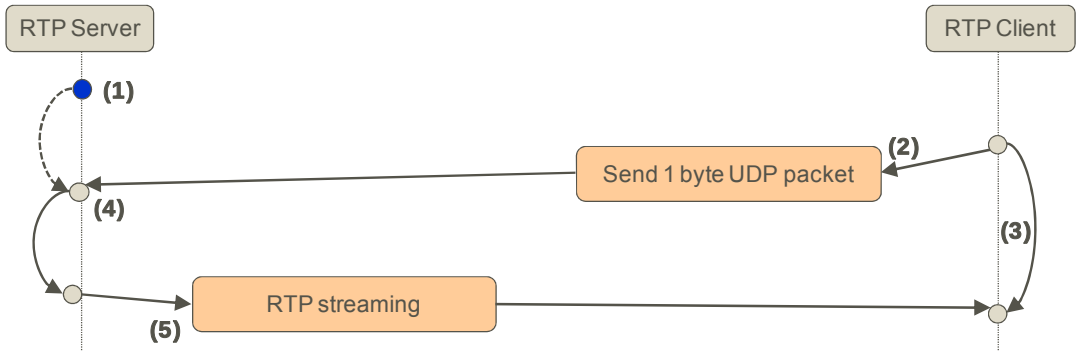


Figure 4: Sequence to Establish the RTP Connection

7.4.3 RTP Server within MirrorLink Client

When the MirrorLink Client takes the role of the MirrorLink RTP Server, the MirrorLink Client shall use the destination IP address and port number from the UPnP *LaunchApplication* action's response. Therefore, there is no need to wait for the 1 byte UDP packet.

The MirrorLink Client's RTP server will start RTP streaming to the RTP client, when audio stream becomes available.

The following sequences show how RTP streaming for a MirrorLink Client based RTP server works:

1) UPnP Control Point: Invoke UPnP *LaunchApplication* action for the respective RTP Client on the MirrorLink Server.

2) UPnP Server: Return IP address and port number for the RTP client as a reply for the *LaunchApplication* request action.

3) RTP Client: Get ready for getting RTP streaming.

4) RTP Server: Starts RTP streaming; if no audio data available, RTP Server should send a single RTP packet, without data payload.

7.5 Client and Server Implementation

For each audio stream, the audio client should do local buffering of the incoming RTP packets and should start playing the audio, if the buffer is filled with a reasonable number of packets. This will allow compensating for potential jittering.

The audio client shall compensate potential frequency differences between the server and client audio sampling rate.

The audio server is responsible for maintaining long term accuracy of the audio clock. For example, if a 48 kHz audio stream is sent over 10 seconds the RTP server should make sure that all data are delivered around 10 seconds' duration with minimal error. Note that each packet can arrive at different intervals depending on the network load, and a reasonable amount of buffering should solve this problem. The audio client is expected to use its own audio clock to play the received audio stream, and it is the audio client's responsibility to compensate potential frequency differences between the server and client audio clock. When an audio server supplies audio stream in much faster frequency than audio client can handle, the audio client can either drop some packets or adjust its playback frequency. It is up to client to decide which method to choose.

It is expected that the audio is played immediately, without any delays, besides the buffering delay. No specific synchronization between the audio stream and the VNC based display updates is provided.

The RTP client should provide the following RTP streaming parameter to the RTP server, to allow uninterrupted audio playback, even if the audio transfer is subject to network jitter:

1) Initial Playback Latency (IPL) [samples]

   If the audio buffer has been empty, the RTP client shall start the audio playback only when the audio buffer is filled with IPL audio samples (the IPL value is based on RTP payload type 99).

   For other payload types, the new IPL is calculated according the following formula:

$$IPL_{payloadType} = floor\left(IPL_{payloadType=99} \cdot \frac{f_{payloadType}}{48kHz} \right)$$

   The default IPL value is 4 800.

   The RTP Client may use a separate low-latency buffer for conversational audio, rather than the application audio buffer as defined with *audioIPL*. Conversational audio over RTP shall use the RTP header extension with *appCategory* = 0xF0000020 (Conversational Audio). Voice Command audio over RTP shall use the RTP header extension with *appCategory* = 0xF0000010 (Voice Command Engine).

NOTE: The RTP Client should pause the audio when RTP packet with Marker bit set to 1 received. The RTP Client should discard the audio buffer content, if audio playback cannot be started or resumed within 1s after receiving the first RTP packet.

   The IPL value shall provide sufficient cover to include any potential delay, originating from the MirrorLink Client's need to ramp-up the MirrorLink audio chain.

2) Maximum Payload Length (MPL) [samples]

   The RTP client shall be able to store MPL audio samples into the audio buffer (the MPL value is based on RTP payload type 99).

   For other payload types, the new MPL is calculated according the following formula:

$$MPL_{payloadType} = floor\left(MPL_{payloadType=99} \cdot \frac{f_{payloadType}}{48kHz} \right)$$

   The default MPL value is 9 600.

The RTP server should send RTP packets, on average, at regular time intervals, equal to the time it takes to play them at the RTP client. The RTP server shall not cause a buffer underrun or overflow at the RTP client. The RTP server may initially send a bigger packet to allow for compensation of future network jitter. The RTP server is responsible for compensating for expected network jitter, by ensuring that enough audio samples are transmitted to the client. The audio playback shall be free of pops, crackles and other disturbing noises and shall not break up unexpectedly.

The RTP Server shall set the M-bit to 1 if no further audio is currently available for streaming, i.e. the RTP Server expects the RTP Client's pipeline getting fully emptied, prior potentially resuming the RTP streaming at a later time. This may happen due to e.g. pausing or terminating the audio source.

EXAMPLE: To indicate the end of a navigation voice guidance, if no other audio sources are playing.

NOTE: The M-Bit cannot necessarily be used to identify the end of an individual audio source.

RTP Client should dispose audio content, if the RTP Server suspends audio streaming (M bit set to 1) or changes the RTP payload type, and if the RTP Client has not started the audio playback.

The MirrorLink Client should provide the RTP streaming parameters within the *TmClientProfile* service as defined in [9]. The MirrorLink Server should provide the RTP streaming parameters within the *A_ARG_TYPE_AppList* response as defined in [8].

The MirrorLink Server shall set the system's audio volume to a reasonable level and shall prevent the user from changing the audio volume during a MirrorLink session.

EXAMPLE: Changing the audio volume via the MirrorLink Server's UI or external volume buttons.

NOTE: The present document does not prevent applications, to intentionally provide an audio stream with a low audio volume.

The MirrorLink Server shall only send RTP packets, if they carry real audio content. I.e. the RTP Server shall not send empty RTP packets or RTP packets with a zero-volume audio signal, if no audio content is available. Otherwise some MirrorLink Client will not be able to switch back to its entertainment audio source in that situation.

When an audio source has indicated that it been paused or stopped, the RTP Server shall not stream any additional RTP packets with unknown application category or unknown application identifier, from that audio source (i.e. no potential guard interval is streamed and exposed to the RTP Client). The RTP Server shall set the M-flag to `"1"` in the last RTP packet from that audio source, if no other audio sources are currently playing; in latter case the RTP extension header shall not include references to the paused/stopped audio source.

If available, the MirrorLink Client may switch to local entertainment sources, after the RTP source on the MirrorLink Server has been paused or stopped.

The RTP Client may initially receive RTP packets with application identifier and/or application category set to `0x00` for up to 1 000 ms, in case the MirrorLink application is using an asynchronous mechanism to set the audio context information and to start the audio stream. It is up to the RTP Client to play, to silently ignore or to block those initial RTP packets.

The MirrorLink Server shall prevent any system sound being sent to the MirrorLink Client. Possible implementation restrictions are detailed in the platform specific specifications.

The MirrorLink Server shall not stream any audio to the MirrorLink Client, representing swipe, touch or key press/release sounds from the consumer interacting with the user interface.

Implementation Note:

MirrorLink 1.0 and 1.1 Servers may stream audio, representing swipe, touch or key press/release sounds. In most cases, the consumer is able to manually disable these system sounds.

8 Voice Command Handling

8.1 General

Voice Commands allow the User to interact and control a MirrorLink application using voice, while being in a MirrorLink session. The Voice Command use case require the MirrorLink Client to have access to a microphone, which is used to capture the user's voice and stream the voice samples to the MirrorLink Server, which processes the voice command in its voice command engine. The application will get notified on the voice command. Voice command engines typically operate on a limited set of commands and grammar.

A MirrorLink Server shall support Voice Command (VC), if the MirrorLink Server devices has a VC engine. A MirrorLink Client shall support VC, if the MirrorLink Client has access to a Microphone. The requirements for handling VC are defined in this clause.

NOTE: Older MirrorLink Clients and Server need not support VC, even if a Client has access to a Microphone or a Server has a Voice Command Engine.

A Voice Command session can be initiated from the MirrorLink Client, e.g. the user pushing a push-to-talk button on the steering wheel, or from the MirrorLink Application, e.g. the user pushing a push-to-talk button on the applications user interface. A MirrorLink Client may use a wake-up word to initiate a VC session.

The Voice Command session shall be established only temporarily, i.e. a MirrorLink Server shall not establish a VC session for the purpose of listening for a wake-up word. A MirrorLink Client shall timeout a VC session.

The Voice Command allows for the 3 user experiences, which are specified in detail below.

1) User pushes a Push-to-Talk button on the MirrorLink Client, while the MirrorLink session is in the foreground of the MirrorLink Client screen.

The MirrorLink Client shall initiate the Voice Command session and shall keep the current MirrorLink session in foreground, unless safety related features, prevent the MirrorLink Client keeping the MirrorLink session in the foreground.

MirrorLink Servers, supporting Voice Commands, shall support this scenario. MirrorLink Clients shall support it, if they have access to a push-to-talk button.

2) User pushed a Push-to-Talk button on the MirrorLink Client, while the MirrorLink session is in the background of the MirrorLink Client screen.

The MirrorLink Client shall initiate the Voice Command session, if it has determined that the Voice Command is for MirrorLink Server consumption. The MirrorLink Client may bring the MirrorLink session into the foreground.

MirrorLink Servers, supporting Voice Commands, shall support this scenario. MirrorLink Clients may support it.

3) User pushes a Push-to-Talk button on the MirrorLink application user interface, via the MirrorLink Client's display.

The MirrorLink Server shall initiate the Voice Command session. The MirrorLink Client shall then keep the MirrorLink session is foreground, while the VC session is active, unless safety related or other higher priority features (e.g. incoming call, navigation prompts), prevent the MirrorLink Client keeping the MirrorLink session in the foreground.

This scenario applies, if an application wants to initiate a voice interaction, e.g. in response to an incoming event.

MirrorLink Servers and Clients, supporting Voice Commands, shall support this scenario.

The MirrorLink Client shall indicate support for Voice Commands push-to-talk buttons (*ptt*) within its *Client Profile* as defined in [9]. VC support is possible over an RTP backchannel (*vc_rtp*) or via Bluetooth HFP BVRA (*vc_bt_hfp*). The MirrorLink Server shall make voice command and push-to-talk button support available to MirrorLink Applications via the MirrorLink API.

8.2 VC over RTP with an Established VNC Session

MirrorLink Servers and Clients, supporting Voice Commands, shall support it over an RTP back channel with an established VNC session. The RTP back channel shall be established with the establishment of the MirrorLink Session.

Implementation Note:

MirrorLink 1.1 or 1.2 Clients may establish the RTP back channel later, when a Voice Command session is getting established.

The MirrorLink Server shall support RTP payload type 100 for Voice Command Audio. The MirrorLink Client shall use RTP payload type 100 for Voice Command Audio, unless the MirrorLink Server does not support it (for MirrorLink 1.1 or 1.2 Servers). The MirrorLink Server shall support RTP payload type 99 for Voice Command Audio from MirrorLink 1.1 and 1.2 Clients.

Most Voice Command engines are natively using payload type 100. Voice Command audio shall not be mixed with another audio stream within the same RTP packet.

The Device Status Voice Command flag indicates the start and end of a VC session, whereas the Device Status Microphone flag indicates the start and end of individual Voice commands. In case of a Voice Command session, consisting of a single Voice Command, both flags can be set at the same time. The basic flow is shown in Figure 5.

Figure 5: Voice Command Session over RTP

A Voice Command session can be initiated either from the MirrorLink Client or from the MirrorLink Server, as described in more detail below.

In case the Voice Command session triggered from the MirrorLink Client, e.g. the user pushing a Push-to-Talk button on the steering wheel, the MirrorLink Client shall request a VC Session sending VNC *DeviceStatusRequest* message with *VoiceCommand* enabled. The MirrorLink Server shall respond with a VNC *DeviceStatus* message with *VoiceCommand* enabled, if VC is supported and available. Otherwise it shall respond with a VNC *DeviceStatus* message with *VoiceCommand* disabled. Once the Voice Command Engine is ready to receive the voice command, the MirrorLink Server shall send a VNC *DeviceStatus* message with *MicrophoneInput* enabled. The message sequence is shown in Figure 6.

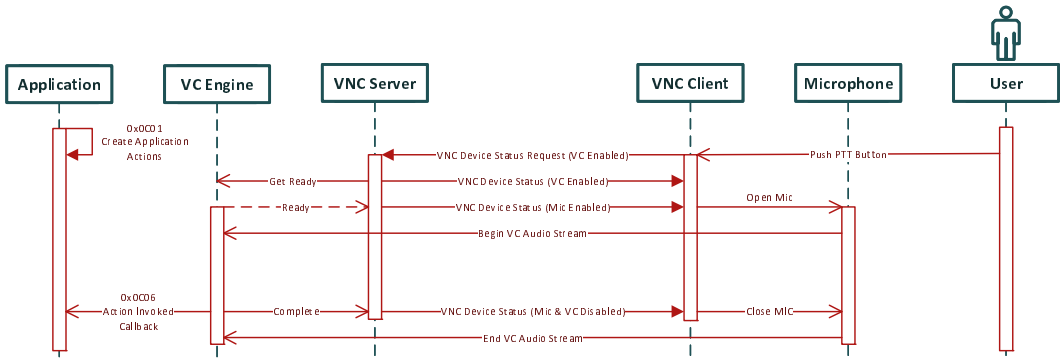


Figure 6: Message Sequence for MirrorLink Client triggered VC

The VNC Client may enable the Microphone as well in its initial VNC Device Status Request message. The VNC Server may enable Voice Command and Microphone within the same VNC Device Status message.

In case the Voice Command session triggered from the MirrorLink Server, e.g. the pushing a button on the MirrorLink application's user interface, the MirrorLink Server shall send the VNC *DeviceStatus* message with *VoiceCommand* enabled. Once the Voice Command Engine is ready to receive the voice command, the MirrorLink Server shall send a VNC *DeviceStatus* message with *MicrophoneInput* enabled. The message sequence is shown in Figure 7.

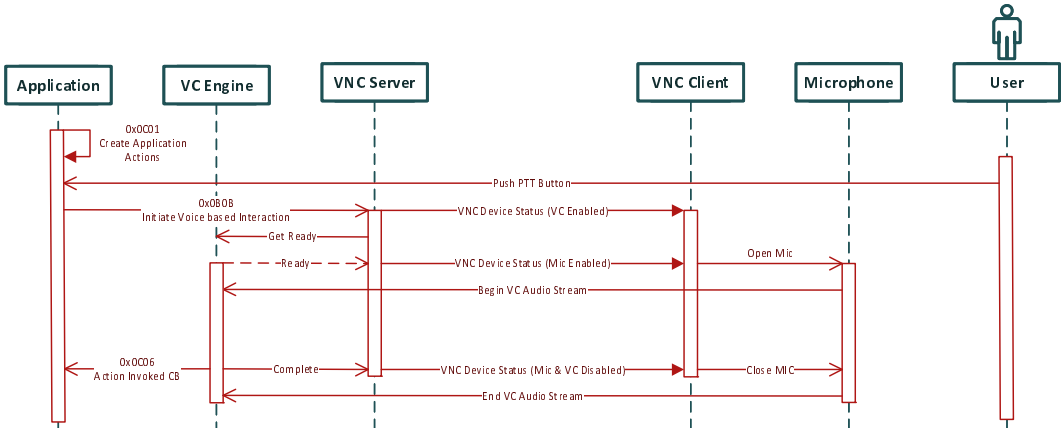


Figure 7: Message Sequence for MirrorLink Server triggered VC

The MirrorLink API references are provided as an indication.

Upon reception of the VNC *DeviceStatus* message with *MicrophoneInput* enabled, the MirrorLink Client shall open the microphone and start the then RTP streaming, as soon as voice command input is available. While the VC session is active, the MirrorLink Client shall send Voice Command audio with RTP header extension equal `0xF0000010`. In case the Microphone is currently not available, e.g. due to safety related considerations, the MirrorLink Client shall send a VNC *DeviceStatusRequest* message with *MicrophoneInput* disabled, as shown below in Figure 8.

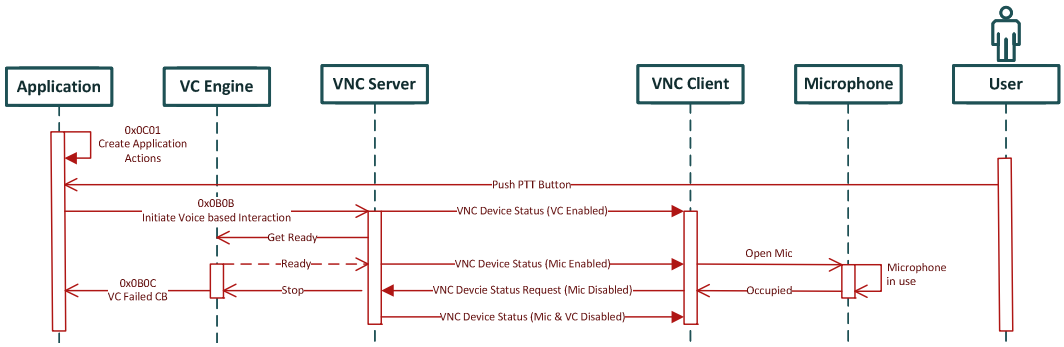


Figure 8: Message Sequence for VC with Occupied Microphone

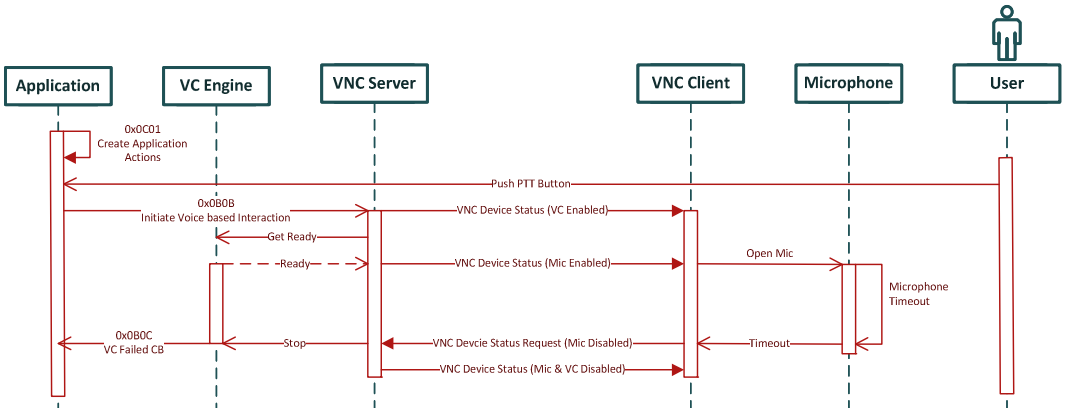


Figure 9: Message Sequence for VC with MirrorLink Client Timeout

An established Voice Command session is terminated from the MirrorLink Server. The MirrorLink Server shall send a VNC *DeviceStatus* message with *MicrophoneInput* and *VoiceCommand* disabled. The MirrorLink Server can disable these flags either within a single VNC *DeviceStatus* message, or within two separate messages (the *MicrophoneInput* shall be disabled first though).

A Voice Command session may be terminated from the MirrorLink Client, either when the Microphone is not available, as shown in Figure 8, or in case of a timeout condition, as shown in Figure 9. Additionally, the user may terminate the VC session as well. In all cases, the MirrorLink Client shall send a VNC *DeviceStatusRequest* message with *VoiceCommand* disabled. The MirrorLink Server shall respond with a VNC *DeviceStatus* message with *MicrophoneInput* and *VoiceCommand* disabled. The termination of the Voice Command session is communicated back to the application.

The MirrorLink Server may terminate a Voice Command session due to a timeout condition as well, as shown in Figure 10.

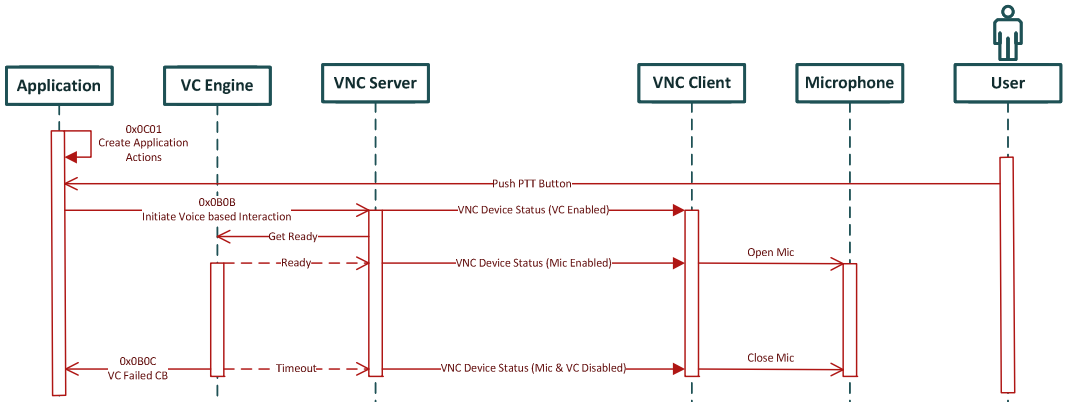


Figure 10: Message Sequence for VC with MirrorLink Server Timeout

The VC engine on the MirrorLink Server may break a Voice Command session into individual VC dialogs, sending a VNC *DeviceStatus* messages first with the *MicrophoneInput* flag disabled and later with enabled. The Dialog phase is repeated then, as shown in Figure 11.

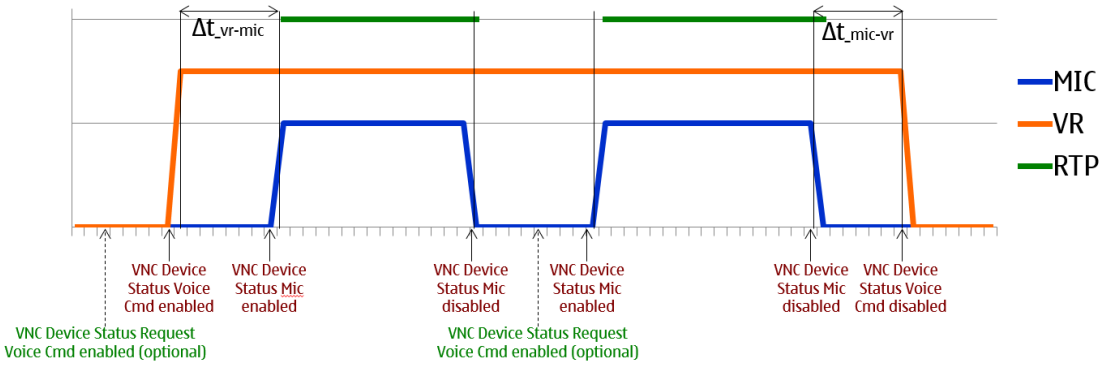


Figure 11: Voice Command Dialog over RTP

The MirrorLink Server should send VNC *DeviceStatus* with the *VoiceCommand* flag enabled together with or before *MicrophoneInput* flag enabled ($\Delta t\_vr\text{-}mic \geq 0$).

The MirrorLink Server should send VNC *DeviceStatus* with the *VoiceCommand* flag disabled together with or after *MicrophoneInput* disabled ($\Delta t\_mic\text{-}vr \geq 0$).

A VC session may take place during a Phone Call Session, as shown below in Figure 12.

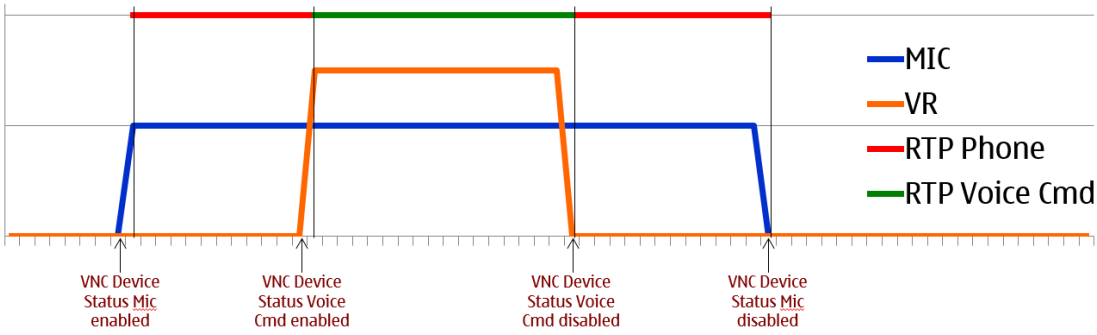


Figure 12: Voice Command Session over RTP with ongoing Phone Call

8.3 VC over RTP with an Established WFD Session

MirrorLink Servers and Clients, supporting Voice Commands, shall support it over an RTP back channel with an established WFD session.

The described RTP behavior, as described in clause 8.2, shall apply to a MirrorLink over WFD session in the same way, using the respective UIBC messages instead the VNC messages.

8.4 VC over RTP without a VNC/WFD Session

Voice Commands over an RTP back channel, without an established VNC or WFD session is not possible.

User may need to first launch an application, or bring an already running application into the foreground on the MirrorLink Client. Alternatively, applications may send a UPnP notification, to have the MirrorLink Client establish a VNC or WFD session.

8.5 VC over Bluetooth HFP BVRA

MirrorLink Servers and Clients, Voice Commands, may support it over Bluetooth HFP BVRA.

In case Bluetooth HFP with BVRA support is established within MirrorLink, Voice Command capabilities shall provide the same capabilities as described in clause 6.1.

In case Bluetooth HFP with BVRA support is established outside MirrorLink, Voice Command capabilities may be limited to HFP use cases.

9 Interoperability with Bluetooth

9.1 General

Interoperability (IOP) of the MirrorLink RTP streaming audio framework with Bluetooth audio profiles shall be given. Bluetooth can be used to replace missing or existing RTP audio streaming functionality as defined in this MirrorLink specification.

The MirrorLink Server shall at least distinguish between speech (phone call) and application audio (media player, navigation directions, etc.).

The MirrorLink Server shall support BT HFP alongside MirrorLink, if it supports phone call or voice control functionality.

NOTE: The MirrorLink Server cannot assume the client supports bi-directional RTP streaming.

A driver-facing MirrorLink Client shall support telephony functionality over Bluetooth HFP alongside MirrorLink.

The Bluetooth HFP connection shall be offered or automatically reconnected when the consumer establishes the MirrorLink session within park and drive mode. Initial Bluetooth pairing may require manual steps from the consumer.

9.2 Bluetooth Profiles relevant for MirrorLink

Bluetooth Headset Profile (BT HSP) [3**]**

Bluetooth Headset profile (BT HSP) shall not be used in MirrorLink.

Bluetooth Hands-Free Profile (BT HFP) [2**]**

If Bluetooth Hands-free profile (BT HFP) is used for voice call together with RTP streaming for application audio, it is the responsibility of the MirrorLink Server to take control of audio link (SCO) when phone call is active: The MirrorLink Server will take care of both opening and closing the SCO audio link. The MirrorLink Client will accept the setup of the SCO audio link.

NOTE: There is no difference for the head-unit, whether the audio link is controlled from a phone application (without MirrorLink being active) or from the MirrorLink Server.

It is the MirrorLink Client's responsibility to make sure that the SCO audio link is opened as soon as possible when requested from the MirrorLink Server. Except the explicit MirrorLink Server's responsibility for the audio link control, Bluetooth Hands-Free Profile 1.5 specification will be followed.

Bluetooth Advanced Audio Distribution Profile (BT A2DP) [4**]**

Bluetooth A2DP may optionally be used to stream application audio from the device to the MirrorLink Client. If RTP streaming is available, it is not RECOMMENDED to use BT A2DP for application audio.

9.3 Interoperability States –MirrorLink Server Perspective

Within the MirrorLink context, the following Interoperability (IOP) states with Bluetooth (BT) can be distinguished, as shown in Figure 13 from the CE MirrorLink Server, i.e. the CE device perspective.

- MirrorLink Server has established neither any MirrorLink and nor any legacy Bluetooth connections (**State: None**)

NOTE: Legacy Bluetooth connections are handled outside MirrorLink.

- MirrorLink Server has established a legacy Bluetooth connection to the vehicle head-unit, but no MirrorLink connection (**State: BT to HU**)
- MirrorLink Server has established a legacy Bluetooth connection to a head-set (**State: BT to HS**)
- MirrorLink Server has established a MirrorLink connection to the MirrorLink Client (**State: ML to HU**)
- MirrorLink Server has established a MirrorLink connection and a legacy Bluetooth connection to the MirrorLink Client (**State: ML to HU + BT to HU**)
- Mobile device has established a Bluetooth connection to a head-set in addition to a MirrorLink connection (**State: ML to HU+ BT to HS**)

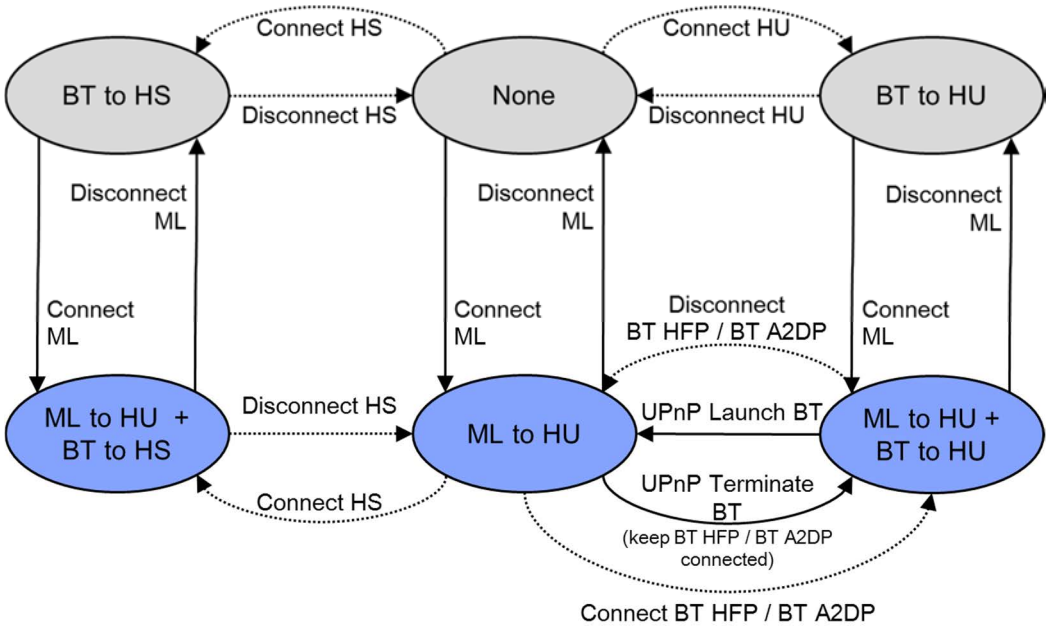


Figure 13: Bluetooth / MirrorLink IOP States – MirrorLink Server Perspective

The transitions between the different IOP states shall be followed from actions with regard to the head-set (HS) or head-unit (HU) Bluetooth profiles or the RTP streaming. These actions are given in Table 12. Transitions marked in dotted lines, are outside the scope of the present document and given for completeness only.

Table 12: IOP Transition (from MirrorLink Server perspective)

Current State	Action	Next State	Immediate Activity for MirrorLink Server Connectivity			
			HS BT HFP	HU BT HFP	HU BT A2DP	HU RTP stream
None	Connect ML	ML to HU	-	Connect (*)	Connect (*)	Connect (*)
	Connect HS	BT to HS	Connect	-	-	-
	Connect HU	BT to HU	-	Connect (**)	Connect (**)	-
BT to HS	Connect ML	ML to HU + BT to HS	Keep	Reject	Reject	Connect (*)
	Disconnect HS	None	Disconnect	-	-	-
BT to HU	Connect ML	ML to HU + BT to HU	-	Keep	Keep	Connect (*)
	Disconnect BT	None	-	Disconnect	Disconnect	-
ML to HU	Disconnect ML	None → BT to HU	-	Keep	Keep	Disconnect
	Connect BT	ML to HU + BT to HU	-	Connect (**)	Connect (**)	Keep
	UPnP Terminate BT	ML to HU + BT to HU	-	Keep	Keep	Keep
	Connect HS	ML to HU + BT to HS	Connect	Disconnect	Disconnect	Keep
ML to HU + BT to HU	Disconnect ML	BT to HU	-	Keep	Keep	Disconnect
	UPnP Launch BT	ML to HU	-	Keep	Keep	Keep
	Disconnect BT	ML to HU	-	Disconnect	Disconnect	Keep
ML to HU + BT to HS	Disconnect HS	ML to HU	Disconnect	Connect (*)	Connect (*)	Keep
	Disconnect ML	BT to HS	Keep	-	-	Disconnect

(*) If selected during MirrorLink UPnP negotiation
(**) If selected during legacy BT pairing

The connection to / disconnection from a wired audio accessory should be treated the same way as the connection / disconnection from a Bluetooth head-set (BT HS).

9.4 Interoperability States –MirrorLink Client Perspective

Within the MirrorLink context, the following Interoperability (IOP) states with Bluetooth (BT) can be distinguished, as shown in Figure 14 from the MirrorLink Client, i.e. Head-Unit perspective.

- MirrorLink Client has established neither any MirrorLink and nor any legacy Bluetooth connections (**State: None**)
- MirrorLink Client has established a Bluetooth connection to a mobile device, but no MirrorLink connection (**State: BT to CE1**)
- MirrorLink Client has established a Bluetooth connection to a mobile device, different from CE1 (**State: BT to CE2**)
- MirrorLink Client has established a MirrorLink connection with CE1 (**State: ML to CE1**)
- MirrorLink Client has established a MirrorLink connection and a legacy Bluetooth connection to the CE1 (**State: ML to CE1 + BT to CE1**)
- MirrorLink Client has established a Bluetooth connection to CE2 in addition to a MirrorLink connection with CE1 (**State: ML to CE1 + BT to CE2**)

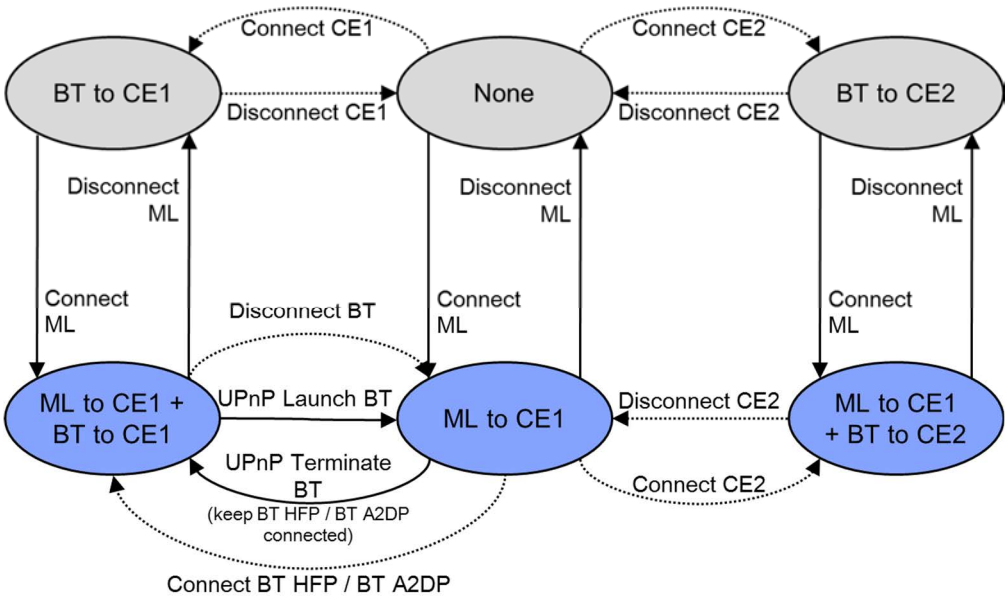


Figure 14: Bluetooth / MirrorLink IOP States – MirrorLink Client Perspective

The transitions between the different IOP states shall be followed from actions with regard to the CE1 and CE2 Bluetooth profiles or the RTP streaming. These actions are given in Table 13. Transitions marked in dotted lines, are outside the scope of the present document and given for completeness only.

Table 13: IOP Transition (from MirrorLink Client perspective)

Current State	Action	Next State	Immediate Activity for MirrorLink Client Connectivity			
			CE2 BT HFP	CE1 BT HFP	CE1 BT A2DP	CE1 RTP stream
None	Connect ML	ML to CE1	-	Connect (*)	Connect (*)	Connect (*)
	Connect CE2	BT to CE2	Connect	-	-	-
	Connect CE1	BT to CE1	-	Connect (**)	Connect (**)	-
BT to CE2	Connect ML	ML to CE1 + BT to CE2	Keep	Reject	Reject	Connect (*)
	Disconnect CE2	None	Disconnect	-	-	-
BT to CE1	Connect ML	ML to CE1 + BT to CE1	-	Keep	Keep	Connect (*)
	Disconnect CE1	None	-	Disconnect	Disconnect	-
ML to CE1	Disconnect ML	None → BT to CE1	-	Keep	Keep	Disconnect
	Connect BT	ML to CE1 + BT to CE1	-	Connect (**)	Connect (**)	Keep
	UPnP Terminate BT	ML to CE1 + BT to CE1	-	Keep	Keep	Keep
	Connect CE2	ML to CE1 + BT to CE2	Connect	Disconnect	Disconnect	Keep
ML to CE1 + BT to CE1	Disconnect ML	BT to CE1	-	Keep	Keep	Disconnect
	UPnP Launch BT	ML to CE1	-	Keep	Keep	Keep
	Disconnect BT	ML to CE1	-	Disconnect	Disconnect	Keep
ML to CE1 + BT to CE2	Disconnect CE2	ML to CE1	Disconnect	Connect (*)	Connect (*)	Keep
	Disconnect ML	BT to CE2	Keep	-	-	Disconnect

(*) If selected during MirrorLink UPnP negotiation
(**) If selected during legacy BT pairing

The selection of the audio link is specified within clause 4.

10 Audio Signal Processing Configuration

10.1 Conversational and Telephony-based Audio

In order to optimize perceived audio quality during conversational audio sessions in MirrorLink, clients and servers should conform to audio performance specifications and behaviours as defined in Recommendation ITU-T P.1100 [i.3]. To achieve this, the MirrorLink Server should disable all audio signal processing functions supported when a MirrorLink connection exists. The MirrorLink Client should perform all audio signal processing functions in the system for conversational audio.

10.2 Bluetooth HFP Noise Reduction/Echo Cancellation

In HFP specification, all audio signal processing in the MirrorLink Server should be disabled in correlation with the HFP requirements specifying AG Noise Reduction/Echo Cancellation.

NOTE: It is assumed that a MirrorLink Server always perform the role of Hands-Free Profile Audio Gateway, as this is the primary way to provide hands-free calling with an in-vehicle infotainment system.

The server should re-enable audio signal processing functions upon audio transfer to the MirrorLink Server device, or disconnection of HFP.

The MirrorLink Client should implement HFP feature to disable audio signal processing, and implement appropriate local audio processing for conversational and telephony audio.

10.3 RTP-based Conversational Audio

If a MirrorLink Client establishes an RTP connection for call or conversational audio usage, the MirrorLink Server should disable all local audio signal processing. The server should re-enable audio signal processing functions upon disconnection of RTP for conversational audio (i.e. call audio transfer to the MirrorLink Server device), or internal detection of idle call state.

If the MirrorLink Client establishes an RTP connection for conversational audio usage, it should implement appropriate local audio processing for conversational and telephony audio.

The RTP Server and Client should follow Recommendation ITU-T G.114 [i.2] for conversational audio.

10.4 RTP-based Voice Command Audio

For Voice Command processing, over RTP the MirrorLink Client shall optimize its local audio signal processing for this use case.

11 Latency Switching Sources

11.1 General

Latency Switching Sources (LSS) describe the problem that switching between different audio sources on a MirrorLink Client can take time. The audio switch has an unspecific latency. This clause describes how a MirrorLink application can be sure that it is audible via the MirrorLink Client, i.e. the switching has been done. Additionally, the respective requirements for the MirrorLink Client and Server are defined.

11.2 Protocol Behaviour of Latency Switching Sources

An application, planning to send an audio stream, shall provide the intended Audio Context Information (MirrorLink API call `0x0803`) to the MirrorLink Server. The MirrorLink Server shall send a silent audio stream, tagged with the provided audio context, in case no other MirrorLink audio source is playing; otherwise the MirrorLink Server shall add the provided audio context information to the already running RTP stream.

The MirrorLink Client shall decide, based on the audio context information within the RTP stream's audio category information and based on the state of the local audio, as defined in clause 7.2, whether to start the MirrorLink audio playback:

MirrorLink Client intends to **playback** the MirrorLink Audio:

1) MirrorLink Client shall send an *Audio Unblocking Notification* message as soon as MirrorLink Client's speaker is ready to receive MirrorLink audio. This message shall include the *application id* of the requesting application, and the *Blocking Reason* set to `0x00`.

2) MirrorLink Server shall send the *Audio In Foreground* callback (MirrorLink API call `0x0808)` to the respective application.

3) MirrorLink Application shall start the audio playback.

MirrorLink Client intends to **not playback** the MirrorLink Audio:

1) MirrorLink Client shall send an *Audio Blocking Notification* message. This message shall include the *application id* of the requesting application and the *Blocking Reason* set to a non `0x00` value. Allowed values are defined in [11].

2) MirrorLink Server shall send the *Audio Blocking* callback (MirrorLink API call `0x0804`) to the respective application.

3) MirrorLink Application shall not start the audio playback.

In case Audio from MirrorLink application A is already playing from the MirrorLink Client's speaker, no additional latency is introduced, when a second MirrorLink application B is starting its playback as well. I.e. the MirrorLink Server shall immediately respond with a locally generated *Audio In Foreground* callback (`0x0808`). A MirrorLink application may use the *Audio In Foreground* MirrorLink API call (`0x0807`) to determine whether MirrorLink audio is already playing from the MirrorLink Client speaker. Note that even if MirrorLink audio is already playing, the MirrorLink Client may still block the new MirrorLink audio stream due to different priorities.

A MirrorLink Application, intending to stop an audio stream, shall stop the audio playback first, and then shall clear the *Audio Context Information* (MirrorLink API `0x0803`), setting the *Audio Content* to `false` and *Audio Category* to `"0"`. The MirrorLink Server shall indicate the end of the MirrorLink audio stream enabling the M-Flag, in case no other MirrorLink Audio stream is playing.

11.3 Timing Behaviour of Latency Switching Sources

Switching between a MirrorLink audio source and a local audio source on a MirrorLink Client will take time, time which is unknown in advance to the MirrorLink Client and Server, as it can dependent on the internal state of the audio system.

For MirrorLink applications it is important though, to understand the time, a MirrorLink Client will need from receiving an RTP stream with new audio context information, until responding with either an *Audio Blocking* or *Audio Unblocking* message. This time is called *Latency Switching Source* (LSS) time. The following two properties for the *LSS* time are defined:

LssMax Defines the **maximum** time the MirrorLink Client needs for making the MirrorLink audio stream audible via the MirrorLink Client's speakers.

NOTE: This time does not include the Initial Play Latency (IPL) as defined in clause 7.5.

LssMax shall be less 1 500 ms (1,5 s). MirrorLink Audio shall be blocked, if switching to it is not possible within *LSS Max* time.

LssAvg Defines the **average** time the MirrorLink Client needs for making the MirrorLink audio stream audible via the MirrorLink Client's speakers.

NOTE: This time does not include the Initial Play Latency (IPL) as defined in clause 7.5.

The value is implementation specific, and should represent the most reasonable switching situations. *LssAvg* shall be less or equal *LssMax*.

The MirrorLink Client shall provide the *LssMax* and *LssAvg* values within the UPnP Client Profile. The values shall match production level systems.

The MirrorLink Server shall make provided *LssMax* and *LssAvg* values available to MirrorLink Applications via the MirrorLink API call `0x010A`. A callback functionality shall be provided through the MirrorLink API callback `0x010B`. The MirrorLink Server shall set both values to `"-1"` in case no information has been provided from the MirrorLink Client.

11.4 Use Cases for Latency Switched Sources

Based on above requirements, the following use cases are relevant:

Blocking MirrorLink Server Audio with no presence of MirrorLink Client Local Audio

The message sequence diagram in Figure 15 shows the application setting the audio context. The MirrorLink Server is sending a silent audio stream, which is responded by a VNC *Audio Blocking* message from the MirrorLink Client. After receiving the blocking callback, the MirrorLink application clears the audio context.

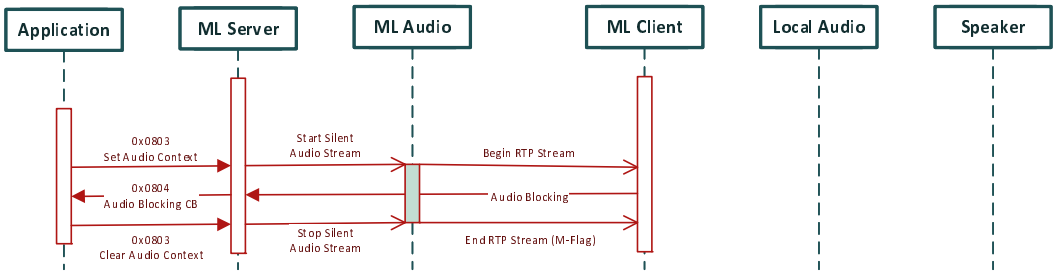


Figure 15: Blocking of MirrorLink Audio with no presence of Local Audio

Unblocking MirrorLink Server Audio with no presence of MirrorLink Client Local Audio

The message sequence diagram in Figure 16 shows the application setting the audio context. The MirrorLink Server is sending a silent audio stream, which is responded by a VNC *Audio Unblocking* message from the MirrorLink Client. After receiving the unblocking callback, the MirrorLink application starts the audio streaming. Finally, the audio is terminated with enabling the M-flag and clearing the audio context.

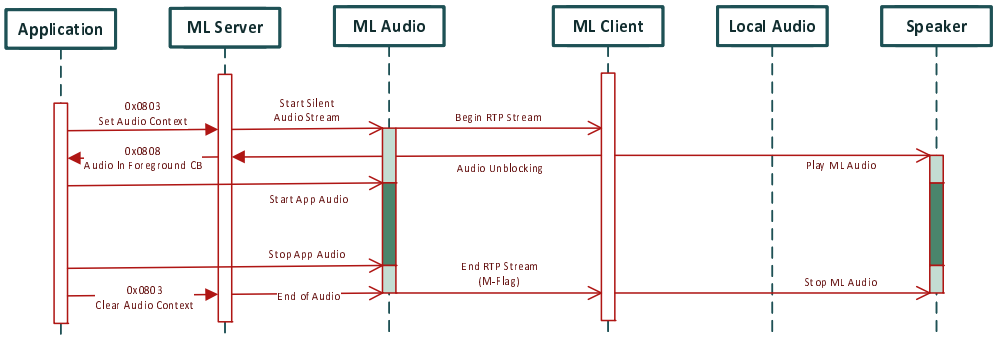


Figure 16: Unblocking of MirrorLink Audio with no presence of Local Audio

Blocking MirrorLink Server Audio with presence of MirrorLink Client Local Audio

The message sequence diagram in Figure 17 shows the application setting the audio context. The MirrorLink Server is sending a silent audio stream, which is responded by a VNC *Audio Blocking* message from the MirrorLink Client. After receiving the blocking callback, the MirrorLink application clears the audio context.

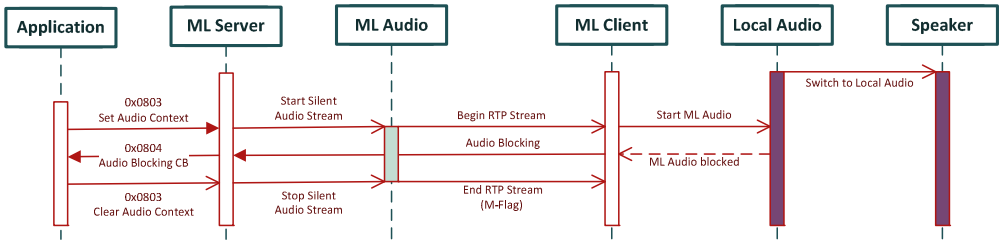


Figure 17: Blocking of MirrorLink Audio with presence of Local Audio

Unblocking MirrorLink Server Audio with presence of MirrorLink Client Local Audio

The message sequence diagram in Figure 18 shows the application setting the audio context. The MirrorLink Server is sending a silent audio stream, which is responded by a VNC *Audio Unblocking* message from the MirrorLink Client, after the audio source has been switched from the Local Audio to the MirrorLink audio (LSS time). After receiving the unblocking callback, the MirrorLink application starts the audio streaming.

Finally, the audio is terminated with enabling the M-flag and clearing the audio context. The Local Audio resumes.

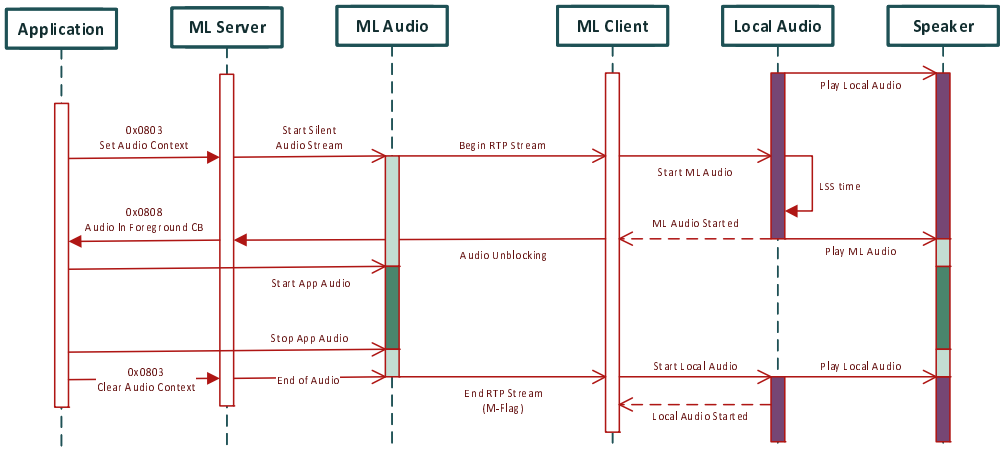


Figure 18: Unblocking of MirrorLink Audio with presence of Local Audio

The MirrorLink Client may not resume the local audio, after the MirrorLink Audio has stopped. The behavior is dependent on the local MirrorLink audio type or user actions.

Unblocking MirrorLink Server Audio with presence of MirrorLink Client Local Audio

The message sequence diagram in Figure 19 shows the application B setting the audio context. The MirrorLink Server is sending a silent audio stream, which is responded by a VNC *Audio Unblocking* message from the MirrorLink Client, after the audio source has been switched from the Local Audio to the MirrorLink audio (LSS time). After receiving the unblocking callback, the MirrorLink application starts the audio streaming.

The procedure happens now for application A, with the exception that MirrorLink Server now locally responding with an unblocking callback for application A, MirrorLink audio is already playing via the MirrorLink Client's speaker.

Finally, the audio of application A and then application B is terminated and clearing the audio context. At the end of the MirrorLink audio stream, the M-flag is enabled. The Local Audio may resume.

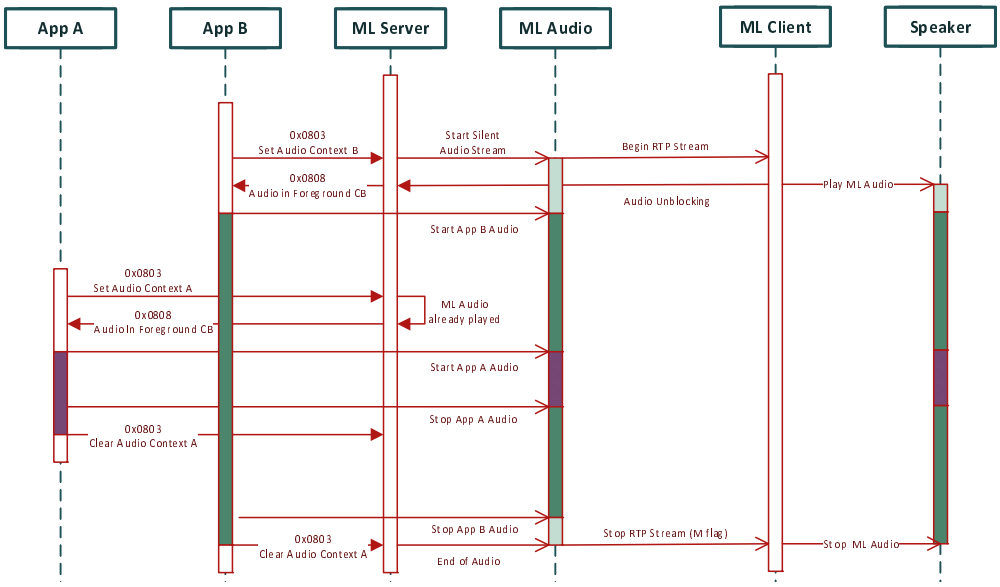


Figure 19: Unblocking of MirrorLink Audio with presence of another MirrorLink Audio

Implementation Note:

Above message sequence diagrams are showing an ideal behavior for the MirrorLink Client's RTP handling. As described in clause 7.5, MirrorLink Clients will implement an RTP buffer, which has to be filled first, as defined by the *Initial Playback Latency* (IPL), prior the audio playback becomes audible from the MirrorLink Client speaker, as shown in the following figure.

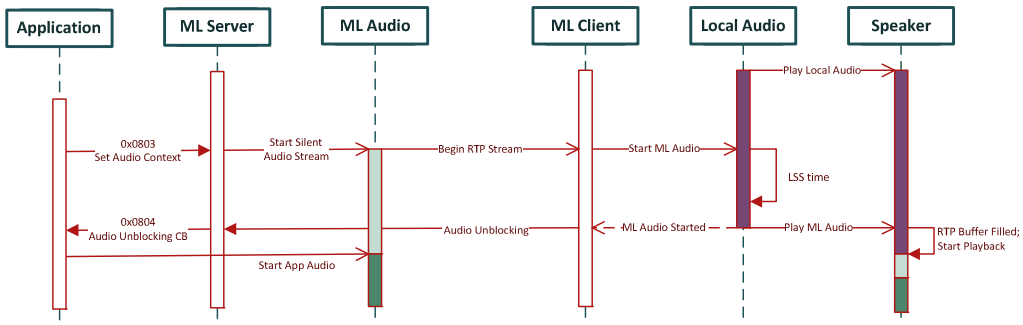


Figure 20: Impact of Initial Playback Latency in Message Sequence

11.5 Backward Compatibility of Latency Switched Sources

Latency Switched Sources have been introduced in MirrorLink 1.3. Previous versions do not support this concept.

Interoperability with MirrorLink 1.0 and 1.1 Servers

MirrorLink 1.0 and 1.1 Servers do not support the specified concept *Latency Switched Sources*. Therefore, the MirrorLink Client shall not send an *Audio Unblocking* messages to indicate access of the MirrorLink audio to the MirrorLink Client's speaker.

Note that above requirement does not apply to the situation where the MirrorLink Client unblocks a previously blocked audio stream.

Interoperability with MirrorLink 1.0 and 1.1 Clients

MirrorLink 1.0 and 1.1 Clients do not support the specified concept *Latency Switched Sources*. Therefore, the MirrorLink Server shall set *LssMax* and *LssAvg* to `"0"` in within the MirrorLink API call `0010A`. The MirrorLink Server shall issue the Audio in Foreground Callback (`0x0808`) as soon as the MirrorLink application has set the audio context (`0x0803`) and the RTP stream has started.

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:	Mark Beckmann, Volkswagen AG
	Matthias Benesch, Daimler AG
	Raja Bose, Nokia Corporation
	Dennis Fernahl, Carmeq (for Volkswagen AG)
	Kari Kostiainen, Nokia Corporation
	Martin Lehner, Jambit
	Keun-Young Park, Nokia Corporation
	Tommy Park, LG Electronics
	Christopher Seubert, Carmeq (for Volkswagen AG)
	Michael Wolf, Daimler AG
	Young-Rang Yoon, LG Electronics

History

Document history		
V1.3.0	October 2017	Publication