ETSI TS 103 544-2 V1.3.1 (2019-10)

TECHNICAL SPECIFICATION

Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 2: Virtual Network Computing (VNC) based Display and Control

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).

CCC is owner of the copyright of the document CCC-TS-010 and/or had all relevant rights and had assigned said rights to ETSI on an "as is basis". Consequently, to the fullest extent permitted by law, ETSI disclaims all warranties whether express, implied, statutory or otherwise including but not limited to merchantability, non-infringement of any intellectual property rights of third parties. No warranty is given about the accuracy and the completeness of the content of the present document.

Reference

RTS/ITS-98-2

Keywords

interface, ITS, PAS, smartphone, USB

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

Copyright Notification

Contents

Intellectual Property Rights 5

Foreword 5

Modal verbs terminology 5

1 Scope 6

2 References 6
2.1 Normative references 6
2.2 Informative references 7

3 Definition of terms, symbols and abbreviations 7
3.1 Terms 7
3.2 Symbols 7
3.3 Abbreviations 7

4 Introduction 8

5 Managing a VNC session 9
5.1 Identifying Remote Applications and the VNC Server 9
5.2 Launching the VNC Session 9
5.3 Intentionally Terminating the VNC Session 9
5.4 Unintentionally Terminating the VNC Session 10
5.5 Testing Considerations 10

6 Traditional VNC Protocol Phases 10
6.1 General 10
6.2 Handshaking Phase 11
6.3 Initialization Phase 12
6.4 Framebuffer Update and Event Phase 13

7 VNC MirrorLink Extension Messages 16
7.1 General 16
7.2 ByeBye Message 17
7.3 Display Configuration Messages 18
7.3.1 General 18
7.3.2 Framebuffer Scaling (VNC Server) 22
7.3.3 Framebuffer Scaling (VNC Client) 23
7.3.4 Handling of Different Framebuffer Aspect Ratios 23
7.3.5 Handling of Server Pixel Aspect Ratios 24
7.3.6 Handling of Application, Framebuffer and Display Orientation 24
7.4 Event Configuration Messages 25
7.5 Event Mapping Messages 29
7.6 Device Status Messages 30
7.7 Content Attestation Messages 34
7.8 Framebuffer Blocking Notification 37
7.9 Audio Blocking Notification 43
7.10 Touch Event 46

8 Additional Encodings and Pseudo Encodings 48
8.1 General 48
8.2 MirrorLink Pseudo Encoding 49
8.3 Context Information Pseudo Encoding 49
8.4 Desktop Size Pseudo Encoding 50
8.5 Scan Line based Run-Length Encoding 51
8.6 VA H.264 Encoding 52
8.6.1 Overview 52
8.6.2 Theory of operation 53

Annex A (normative): Knob Configuration 59

Annex B (normative): Key Event Mapping 60

Annex C (normative): Language Sets 63

C.1 Basic Set Latin-1 63

Annex D (informative): Authors and Contributors 64

History 65

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 2 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 contents of the MirrorLink Server device's screen are transferred to the MirrorLink Client device. The control inputs are transferred from the MirrorLink Client to the MirrorLink Server. Screen copy methods can be used to copy the content of the MirrorLink Server's framebuffer to the MirrorLink Client's display. The copy operation can include rotation or colour conversion. The frame buffer is used as an abstraction layer, allowing any changes to the applications and services running on the mobile device to be avoided. For this purpose, the Virtual Networking Computing (VNC) protocol is used.

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 6143: "The Remote Framebuffer Protocol", March 2011.

NOTE: Available at https://tools.ietf.org/html/rfc6143.

[2] Bluetooth® Specification: "Hands-free Profile", Audio, Telephony, and Automotive Working Group, Revision 1.7.2, January 21, 2019

NOTE: Available at https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=457090.

[3] ETSI TS 103 544-3 (V1.3.1): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 3: Audio".

[4] ETSI TS 103 544-9 (V1.3.1): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 9: UPnP Application Server Service".

[5] ETSI TS 103 544-10 (V1.3.1): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 10: UPnP Client Profile Service".

[6] ETSI TS 103 544-26 (V1.3.1): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink® ; Part 26: Consumer Experience Principles and Basic Features".

[7] X Consortium Standard: "X Window System Protocol", X Version 11, Release 6.9/7.0.

NOTE: Available at ftp://ftp.x.org/pub/X11R7.0/doc/PDF/proto.pdf.

[8] Recommendation ITU-T H.264 (04-2017): "Advanced video coding for generic audiovisual services".

NOTE: Available at https://www.itu.int/rec/T-REC-H.264-201704-S/en.

[9] ISO 639-1: "Codes for the representation of names of languages -- Part 1: Alpha-2 code".

[10] ISO 3166-1: "Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes".

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 (V1.3.1): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 1: Connectivity".

[i.2] US Department of Homeland Security: "Emergency Alerts".

NOTE: Available at https://www.ready.gov/alerts.

[i.3] IANA, Protocol Registries, Remote Framebuffer (RFB) assignments.

NOTE: Available at https://www.iana.org/assignments/rfb/rfb.xhtml.

[i.4] ISO/IEC 10646:2014: "Information technology -- Universal Coded Character Set (UCS)".

[i.5] ETSI TS 103 544-22 (V1.3.1): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 22: Android Specific Specifications enabling AIDL-based MirrorLink® Applications".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms 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 Symbols

Void.

3.3 Abbreviations

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

A2DP	Bluetooth Advanced Audio Distribution Profile
API	Application Programming Interface
ARGB	Alpha-Red-Green-Blue
BT	Bluetooth
BVRA	Bluetooth Voice Recognition Activation

DAP	Device Attestation Protocol
FAR	Framebuffer Aspect Ratio
FBU	FrameBuffer Unit
FEMA	Federal Emergency Management Agency
HSML	High-Spead Media Link
IANA	Internet Assigned Numbers Authority
ISO	International Standards Organization
NAL	Network Abstraction Layer
PAR	Pixel Aspect Ratio
PIN	Personal Identification Number
PPS	Picture Parameter Set
RFB	Remote Framebuffer
RGB	Red-Green-Blue
RLE	Run-Length Encoding
RTP	Real-time Transport Protocol
SPS	Sequence Parameter Set
TCP	Transmission Control Protocol
TV	TeleVision
UDP	User Datagram Protocol
UI	User Interface
UPnP	Universal Plug and Play
URL	Universal Resource Locator
US	United States
USB	Universal Serial Bus
UTF	Unicode Transformation Format
VA	Video Audio
VCL	Video Coding Layer
VNC	Virtual Network Computing

4 Introduction

The Virtual Networking Computing (VNC) uses the Remote Framebuffer Protocol (RFB) as a simple protocol for remote access to any sort of framebuffer-based user interface. The remote endpoint is called the VNC Client, whereas the endpoint driving the framebuffer is called the VNC Server. In the MirrorLink context, the VNC Client resides in the vehicle head-unit (MirrorLink Client) and the VNC Server is in the mobile device (MirrorLink Server). The VNC Client will show the remote display either on the entire local display or on a subset of it, as shown in Figure 1.

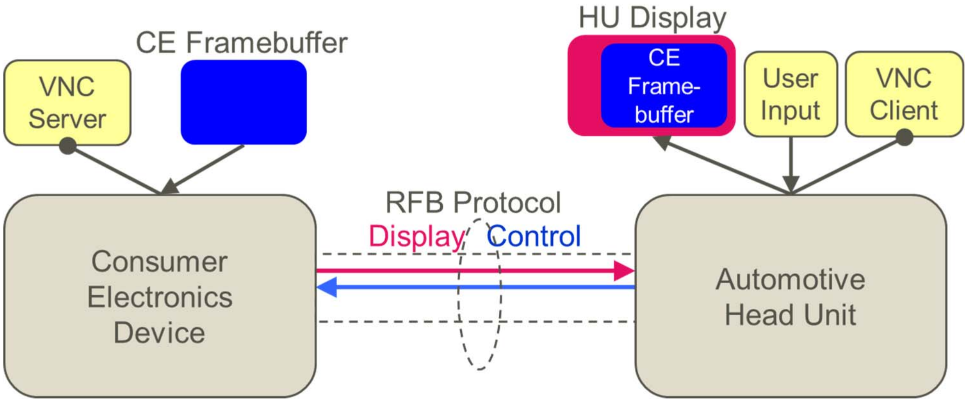

Figure 1: MirrorLink VNC Setup

The command and control input is handled as part of the VNC protocol by key and pointer events. A key or pointer event on the MirrorLink Client will be signalled to the MirrorLink Server via a specific key symbol value, which uniquely identifies the event. The mobile device and/or its application will not necessarily support all possible keys defined. Some applications may even have a dynamic behaviour on the selection of key inputs they expect.

The RFB protocol originates from the desktop computing world and has been designed as a thin client protocol, i.e. it assumes a VNC Client with only a few requirements, and a VNC Server having access to more processing capabilities.

The protocol allows the VNC Client to be as simple as possible. In the MirrorLink context this assumption needs to be reconsidered, as mobile devices are experiencing performance limitations as well.

The MirrorLink Client shall implement the VNC Client functionality.

The MirrorLink Server shall implement the VNC Server functionality.

5 Managing a VNC session

5.1 Identifying Remote Applications and the VNC Server

The identification of remote VNC based applications and of the VNC Server is described in [4].

5.2 Launching the VNC Session

The VNC Server start-up is automatically facilitated via UPnP. The MirrorLink Server's VNC Server shall be running, when launching any application in response to a UPnP TmApplicationServer:1 service *LaunchApplication* action, as defined in [4]. The *LaunchApplication* action shall return with a URL to the VNC Server, hosting the VNC session.

If the returned URL is already used from any established VNC session, this session will continue without any change.

Otherwise a new VNC session shall be established, given the following steps:

1) VNC Server shall listen for the VNC Client to make TCP connection at the provided URL.

2) VNC Client shall make a TCP connection to the provided URL.

3) VNC Server and Client shall initialize the VNC session according to the VNC protocol.

5.3 Intentionally Terminating the VNC Session

A VNC session can be terminated any time from both the VNC Server and Client. This mechanism is not meant to handle error situations, which require immediate action from both the VNC Server and Client (use the unintentional termination mechanisms instead). In particular, intentional termination shall be used, if the VNC session is terminated based on an intended user action resulting in a termination of the VNC session.

The VNC Client shall terminate a VNC session using the VNC *ByeBye* message, as given below:

1) VNC Client shall send a VNC *ByeBye* message. The VNC Client shall not send any further VNC message, after sending the VNC *ByeBye* message. The VNC Client should ignore all incoming VNC messages, after sending a VNC *ByeBye* message.

2) VNC Server shall respond with a VNC *ByeBye* message.

3) VNC Client shall disconnect the TCP connection. The VNC Client should disconnect the TCP connection, if it does not receive a VNC *ByeBye* message back within 5 s.

4) VNC Server should disconnect the TCP connection on detection of the VNC Client TCP disconnect or 5 s after sending the VNC *ByeBye* message, whatever comes first.

The VNC Client shall wait with the VNC *ByeBye* message, if it has an outstanding UPnP *LaunchApplication* action until it has received the corresponding UPnP response. This avoids any potential race condition problems.

The VNC Server shall terminate a VNC session, using the VNC *ByeBye* message, as given below:

1) VNC Server shall send a VNC *ByeBye* message. The VNC Server shall not send any further VNC message, after sending the VNC *ByeBye* message. The VNC Server should ignore all incoming VNC messages, after sending a VNC *ByeBye* message.

2) VNC Client shall disconnect the TCP connection.

3) VNC Server should disconnect the TCP connection on detection of the VNC Client TCP disconnect or 5 s after sending the VNC *ByeBye* message, whatever comes first.

It is up to the MirrorLink Client, whether a new VNC session is launched immediately, after the old one has been terminated, from the VNC Client or Server.

The MirrorLink Client should send a UPnP TmApplicationServer:1 service *TerminateApplication* action for the stand-alone VNC Server, if the VNC Client has previously launched it for the terminated session.

Terminating a VNC session shall not impact the application status of any application on the MirrorLink Server. If the MirrorLink Client decides to re-establish the VNC session, it shall follow the steps given in Clause 5.2.

If MirrorLink Server has intentionally terminated the VNC session, the MirrorLink Client should provide a mechanism to start a VNC session again. If the MirrorLink Client decides to re-establish the VNC session, it shall follow the steps given in Clause 5.2.

In case the MirrorLink Client has intentionally terminated the VNC session, the MirrorLink Client shall wait until it received the VNC *ByeBye* message from the MirrorLink Server (or it ran into the disconnect timeout), prior sending any new UPnP Launch Application message.

In case the MirrorLink Server has intentionally terminated the VNC session, it shall wait until it detects the TCP disconnect (or it ran into the disconnect timeout), prior responding to any new UPnP Application launch action or it shall use a different URL than the previous VNC session.

5.4 Unintentionally Terminating the VNC Session

Unintentional termination of the VNC session may happen any time due to error conditions. In case of unintentional termination of the VNC session, the respective VNC Server or Client will disconnect the TCP connection. The respective counterpart should disconnect as well.

If the MirrorLink Client decides to re-establish the VNC session, it shall follow the steps given in Clause 5.2.

To avoid the VNC Server or Client persisting in a TCP TIME-WAIT time-out loop, as a result of an unintentional active disconnect, the TCP socket should be established using the SO_REUSEADDR option (or similar platform specific variants), allowing the operating system to reuse a port address, even it is currently in the TIME-WAIT state or the VNC Server should use a different, unaffected port number.

5.5 Testing Considerations

If the MirrorLink Client is in a dedicated testing state (as part of the MirrorLink Certification), it shall launch a new VNC session (either initiated automatically or manually from the user), whenever the VNC Server has intentionally terminated the VNC session.

If the MirrorLink Client is in a dedicated testing state (as part of the MirrorLink Certification), it shall launch a new VNC session (either initiated automatically or manually from the user), whenever the VNC Server has unintentionally terminated the VNC session.

6 Traditional VNC Protocol Phases

6.1 General

After the connection between the VNC Server and Client has been established, the VNC protocol processing will start according to the VNC specification. The VNC protocol consists of three main steps:

1) Exchange of handshaking messages. After the handshaking phase, the VNC connection parameters are negotiated and the connection is established.

2) Exchange of initialization messages. After this phase, both ends have agreed on all needed parameter for the following operational phase.

3) VNC Client to Server and VNC Server to Client messages are used to reflect changes of the framebuffer content on the local endpoint and user interaction on the remote endpoint.

These three VNC protocol phases are specified in more detail in the following clause.

6.2 Handshaking Phase

The handshaking phase defines a couple of messages, which are exchanged between the VNC Client and the VNC Server, as shown in the Figure 2. In general, the VNC Server presents its capabilities and the VNC Client selects the best option with regard to its own capabilities.

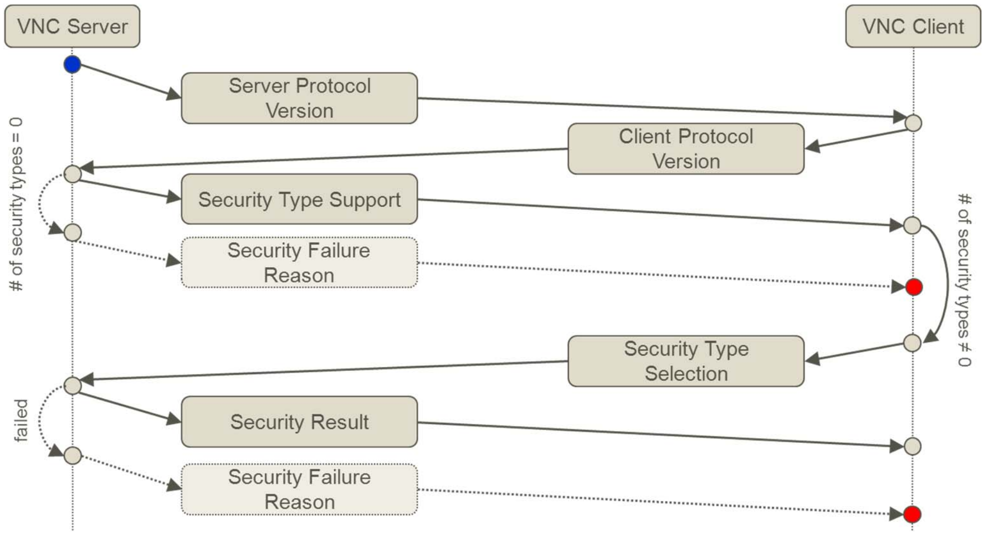

Figure 2: VNC Handshaking Phase

The Table 1 parameters shall be supported from the VNC Client and the Server.

Table 1: Requirements for Handshaking Phase Messages

Message	Origin	Parameter	Mandatory Values
Protocol Version	Server	Max. protocol version	At least 3.8
Protocol Version	Client	Max. protocol version	3.8
Security Type Support	Server	# of security types	(as specified in RFB)
		Security types	1 (None)
Security Type Selection	Client	Security type	(as specified in RFB)
Security Failure Reason	Server	Reason length	(as specified in RFB)
		Reason string	
Security Result	Server	Security status	(as specified in RFB)

Authentication and security are handled outside the VNC protocol on the link-layer and transport-layer. The VNC Client cannot expect the VNC Server to offer additional security or authentication features.

The VNC Client shall disconnect the TCP connection, after receiving a Security Failure Reason from the VNC Server. The VNC Server should disconnect the TCP connection on detection of the VNC Client TCP disconnect or 5 s after sending the VNC *SecurityFailureReason* message, whatever comes first.

The VNC Server and VNC Client shall support a MirrorLink 1.0 compliant counterpart supporting only RFB version 3.7.

6.3 Initialization Phase

The initialization phase defines a couple of messages, which are exchanged between the VNC Client and the VNC Server, as shown in the Figure 3. In general, the VNC Server presents its capabilities and the VNC Client selects the best option with regard to its own capabilities.

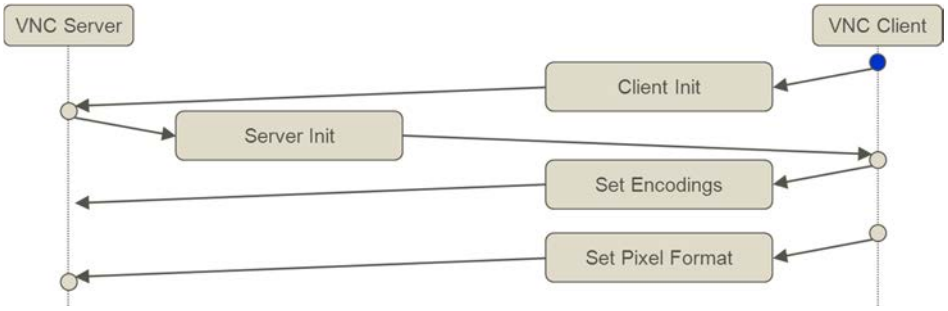

Figure 3: Initialization Phase

The Table 2 parameters shall be supported from the VNC Client and the VNC Server. For the RFB message structure, please refer to the dedicated RFB specification, as given in [1].

Table 2: Requirements for Initialization Phase Messages

Message	Origin	Parameter	Mandatory Values
Client Init	Client	Shared-flag	(as specified in RFB)
Server Init (using native framebuffer configuration)	Server	Framebuffer-width	(as specified in RFB)
		Framebuffer-height	
		Name-length	
		Name-string	
		Bits-per-pixel	
		Depth	
		Big-endian-flag	
		True-colour-flag	
		Red-/Green-/Blue max	
		Red-/Green-/Blue shift	
Set Encodings	Client	Number of encodings	(as specified in RFB)
		Encoding-type	List of required Encodings is given in Clause 8.
Set Pixel Format	Client	Bits-per-pixel	ARGB 888, RGB 565
		Depth	
		Big-endian-flag	
		True-colour-flag	
		Red-/Green-/Blue max	
		Red-/Green-/Blue shift	

The MirrorLink Server shall start in Landscape orientation. The MirrorLink Server may start in Portrait Mode within the VNC *ServerInit* message, but it shall switch to Landscape Mode using the *DesktopSizePseudoEncoding* message, with the first *FramebufferUpdate* message, following the MirrorLink Client's VNC *SetEncodings* message announcing MirrorLink support.

The MirrorLink Server shall support a framebuffer resolution of at least 800 x 480. The actual transmitted framebuffer resolution may be less to preserve the MirrorLink Server framebuffer's aspect ratio on the MirrorLink Client's display, i.e. one framebuffer dimension shall be equal to original framebuffer resolution.

The MirrorLink limits the RFB protocol, as shown in Table 2 with regard to supported colour formats, to allow for efficient implementations. Some more specific recommendations and requirements are given below.

The VNC Client shall not select a pixel format, for which the Server has not indicated support, using the Server Display Configuration VNC extension message. The VNC Server shall support ARGB 888 and RGB 565 pixel formats. The VNC Client shall at least support either ARGB 888 or RGB 565.

The VNC Client shall initially send all supported encodings within a single *SetEncodings* message. The encoding order may be used from the VNC Server as an indication on the VNC Client's priority order (first entry has highest priority). Subsequent *SetEncodings* messages shall not invalidate the use of any previous encoding or pseudo encoding, even if encodings are not repeated. They may change the priority order though. The initial *SetEncodings* message shall be sent prior the first *FramebufferUpdateRequest* message.

NOTE: The VNC Client can include support for other framebuffer encodings, referenced within [i.3], and VNC Servers can use them if supported from both devices.

NOTE: MirrorLink 1.1 or 1.2 devices can support VA H.264 framebuffer encoding, in line with previous note. MirrorLink 1.3 devices cannot expect these implementations to be fully compliant with all requirements defined in Clause 8.6.2 though. Hence, a MirrorLink 1.3 Client may exclude VA H.264 framebuffer encoding from MirrorLink 1.1 and 1.2 Servers.

The VNC Client shall not send a *SetPixelFormat* message, after the first *FramebufferUpdateRequest* message.

6.4 Framebuffer Update and Event Phase

The update and event phase defines a couple of messages, which are exchanged between the VNC Client and the VNC Server. The VNC Server only responds to framebuffer update requests, as shown in Figure 4. No response message is sent to any of the other messages.

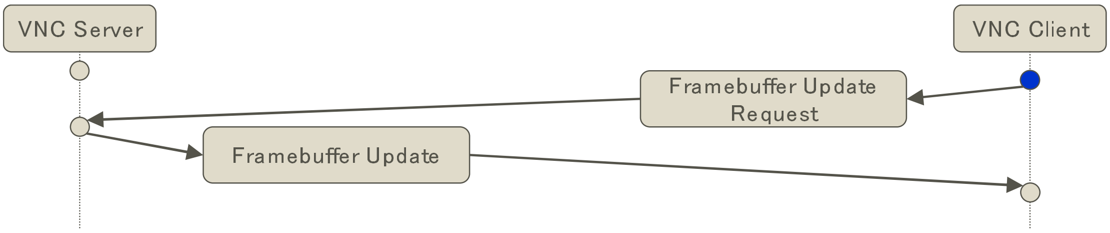

Figure 4: Framebuffer Update Phase

The Table 3 parameters shall be supported from the VNC Client and the VNC Server.

Table 3: Requirements for Framebuffer Update and Event Phase Messages

Message	Origin	Parameter	Mandatory Values
Framebuffer Update Request	Client	Incremental	(as specified in RFB)
		x-position	
		y-position	
		Width	
		Height	
Framebuffer Update	Server	Number-of-rectangles	(as specified in RFB)
		x-position	
		y-position	
		Width	
		Height	
		encoding-type	0 (Raw)

Message	Origin	Parameter	Mandatory Values
Set Colour Map Entries	Server	first colour	(shall not be used during a MirrorLink session)
		number-of-colours	
		Red	
		Green	
		Blue	
Key Event	Client	Down-flag	(as specified in RFB)
		Key	
Pointer Event	Client	Button-mask	(as specified in RFB)
		x-position	
		y-position	
Client Cut Text	Client	Length	(as specified in RFB)
		Text	
Bell	Server	No parameter	(as specified in RFB)
Server Cut Text	Server	Length	(as specified in RFB)
		Text	

NOTE: For the RFB message structure, please refer to the dedicated RFB specification, as given in [1].

The VNC Client can request two types of framebuffer updates, using the incremental flag in the *FramebufferUpdateRequest* message:

- A '0' indicates the VNC Server shall provide a non-incremental update, i.e. the VNC Server shall provide the requested area, independent of whether it has changed or not (Exception: Within a *FramebufferUpdate* message containing a Desktop Size Pseudo Encoding rectangle, the framebuffer data rectangle(s) shall be skipped - see Clause 8.4).
- A '1' indicates the VNC Server should provide an incremental update, i.e. the VNC Server should provide only the area(s) containing all framebuffer changes. The VNC Server should wait before sending a *FramebufferUpdate* message, until the framebuffer content has actually changed. The VNC Server should not continuously send *FramebufferUpdate* messages containing no framebuffer data. In any case, the VNC Client shall support *FramebufferUpdate* messages, where the provided framebuffer width or height equals zero.

NOTE: Those framebuffer updates are not forbidden from the RFB specification specifically. Though some existing VNC Clients, display warnings.

The VNC Client should send the first *FramebufferUpdateRequest* message, after completion of the Display and Event Configuration, as specified in Clauses 7.3 and 7.3.6. The VNC Client shall only request framebuffer data it can handle, without disconnecting the VNC session.

The VNC Server shall not provide a Framebuffer Update outside the rectangular area, which has been requested from the VNC Client. The VNC Server shall send a *FramebufferUpdate* message only in response to a *FramebufferUpdateRequest* message. The VNC Server may ignore *FramebufferUpdateRequest* messages, if multiple messages are sent at the same time. It is recommended that the VNC Client sends only one *FramebufferUpdateRequest* message at a time.

The VNC Client should have a copy of the Server side framebuffer locally available. Therefore, it is recommended that the VNC Client requests incremental framebuffer updates.

The VNC Server shall support at least the following key events from the Latin1 character set:

- `'a' - 'z' (0x0061 – 0x007A)`
- `'A' - 'Z' (0x0041 – 0x005A)`
- `'0' – '9' (0x0030 – 0x0039)`
- `' ' (0x0020) - Space`
- `'!' (0x0021) - Exclamation mark`

- '"' (0x0022) - Quotation mark
- '#' (0x0023) - Number sign
- '$' (0x0024) - Dollar sign
- '%' (0x0025) – Percent sign
- '&' (0x0026) - Ampersand
- ''' (0x0027) - Apostrophe
- '(' (0x0028) – Parenthesis left
- ')' (0x0029) – Parenthesis right
- '*' (0x002A) – Asterisk
- '+' (0x002B) – Plus
- ',' (0x002C) - Comma
- '-' (0x002D) - Minus
- '.' (0x002E) - Period
- '/' (0x002F) - Slash
- ':' (0x003A) - Colon
- ';' (0x003B) - Semicolon
- '<' (0x003C) – Less-than
- '=' (0x003D) - Equal
- '>' (0x003E) – Greater-than
- '?' (0x003F) – Question mark
- '@' (0x0040) - At
- '[' (0x005B) – Bracket left
- '\' (0x005C) – Back slash
- ']' (0x005D) – Bracket right
- '^' (0x005E) – Circumflex
- '_' (0x005F) – Underscore
- '`' (0x0060) - Grave
- '{' (0x007B) – Brace left
- '|' (0x007C) - Bar
- '}' (0x007D) – Brace right
- '~' (0x007E) - Tilde
- (0xFF08) – Backspace
- (0xFF0D) - Return

In case the MirrorLink Server supports other languages, Annex C defines language specific character sets, which shall be supported.

The VNC Server shall ignore all non-supported events. The VNC Server shall not remap any X11 key events (e.g. letter A to letter B), as specified in X11/keysymdef.h from any X Windows installation. The VNC Server shall follow behaviour for supported X11 keys as given in keysymdef.h.

The VNC Server shall support the following ways to interpret a long key-press event:

1) The VNC Client will send a key-press event, and after a longer delay, a key-release event.

2) The VNC Client will send a key-press event, and will continue sending key-press events, until the final key-release event.

The VNC Server shall ignore a key-release event, if no key-press event has been received before for the same key symbol value. The VNC Server shall complete an open key-press event after 5 seconds if no key-release event or no additional key-press event has been received for the same key symbol value. The same behaviour shall apply to pointer events.

NOTE: It is up to the MirrorLink Server to detect a long key-press event based on the implementation-specific value of the time interval between the first key-press event and the key-release event for a specific key value.

The VNC Client should avoid using driver-initiated long key-press events, as MirrorLink will not be able to guarantee safe operation, while driving. E.g. if the driver needs to interrupt an intended long key-press event, the MirrorLink Client event will fall back to a short key-press event, triggering some unintended behaviour.

For short key-press events, the VNC Client shall send a key-release event for every key-press event. The VNC Client should send key-press and key-release events in one TCP packet to avoid mishandling as a long key-press event due to network latency.

In order to send a Unicode/ISO 10646 character within a VNC *KeyEvent* message, the VNC Client shall map the Unicode/ISO 10646 key event range `U0100` to `U10FFFF` to the X11 key symbol value range `0x01000100` to `0x0110FFFF` as specified in Appendix A of [7]. For other Unicode values, found in ISO 10646 / Unicode, the following algorithm shall be used: The X11 key symbol value is the character's Unicode number plus 0x01000000 Appendix A of [7].

NOTE: The Latin-1 characters in the first row of ISO 10646 (U0000 to U00FF) are already represented by key symbol values with the same value.

In order to send Unicode/ISO 10646 characters as UTF-16 characters within a Client or *ServerCutText* message, the VNC Client or Server shall use the following escape sequences within the Text entry.

- `ESC%g (0x1B 0x25 0x67)` Selects UTF-16 character set
- `ESC%@ (0x1B 0x25 0x40)` Returns back to Latin-1

EXAMPLE: The following byte sequences encode a single Greek capital sigma character Σ (0x03A3):

```
0x1B 0x25 0x67                      (select UTF-16 using Latin-1)
0x03 0xA3                           (Greek capital sigma using UTF-16)
0x00 0x1B 0x00 0x25 0x00 0x40       (return to Latin-1 using UTF-16)
```

Any selection of the UTF-16 character set shall be returned to the Latin-1 set within the same Client or *ServerCutText* message. Latin-1 is the default character set.

7 VNC MirrorLink Extension Messages

7.1 General

The existing RFB protocol specification is not sufficient to cover the mobile device - remote car display configuration space. Therefore, extensions to the current protocol are specified in this clause. Extensions are made in compliance with the RFB protocol, introducing a new MirrorLink message type (128).

Under the MirrorLink message type, a couple of additional messages are introduced to the VNC protocol. These can be distinguished via unique extension-types. All extension messages shall use the MirrorLink message type and shall follow the Table 4 fundamental design principle.

Table 4: VNC Extension Type Message Structure

# bytes	Type	Value	Description
1	U8	128	MirrorLink Message-type
1	U8		Extension-type
2	U16	N	Payload length
N	U8 array		Message specific payload data

The VNC Server and Client shall handle MirrorLink extension messages with unknown extension types, by reading the whole message (body and payload) and ignoring it.

The VNC Server and Client shall handle known MirrorLink extension messages, which have been amended in future releases (i.e. the given payload is bigger than the known one), by reading the remaining bytes and ignoring them.

Some VNC extension messages contain bit field entries, marked with "Bit" in the Value Column and a list of possible bit positions underneath, given in the format [n], where n is a valid bit position. A zero-bit position [0] is referring to the least significant bit in the given entry. Bit fields are described as [n:m], with n > m.

All bits, which are not defined within the present document, shall be set to zero, unless otherwise stated.

Some VNC extension messages contain value entries, marked with "Value" in the Value Column and a list of possible values underneath. Only one value can be assigned to the given entry at any given time.

The MirrorLink message type defines the following set of new VNC messages given in Table 5. The table lists the requirement level for supporting those messages for the VNC Server and Client.

Table 5: New Extension Types for MirrorLink Messages

Extension-Type	Message Name	Origin	Server Support	Client Support
0	ByeBye	Server	Shall	Shall
0	ByeBye	Client	Shall	Shall
1	ServerDisplayConfiguration	Server	Shall	Shall
2	ClientDisplayConfiguration	Client	Shall	Shall
3	ServerEventConfiguration	Server	Shall	Shall
4	ClientEventConfiguration	Client	Shall	Shall
5	EventMapping	Server	Shall	Deprecated
6	EventMappingRequest	Client	Shall	Deprecated
7	KeyEventListing	Server	Deprecated	Deprecated
8	KeyEventListingRequest	Client	Deprecated	Deprecated
9	VirtualKeyboardTrigger	Server	Deprecated	Deprecated
10	VirtualKeyboardTriggerRequest	Client	Deprecated	Deprecated
11	DeviceStatus	Server	Shall	Shall
12	DeviceStatusRequest	Client	Shall	Shall
13	ContentAttestationResponse	Server	Should	Should
14	ContentAttestationRequest	Client	Should	Should
16	FramebufferBlockingNotification	Client	Shall	Shall
18	AudioBlockingNotification	Client	Shall	Shall
20	TouchEvent	Client	Should	Should
21	FramebufferAlternativeText	Server	Deprecated	Deprecated
22	FramebufferAlternativeTextRequest	Client	Deprecated	Deprecated

The VNC Client shall disable the key event listing and the virtual keyboard trigger support in the *ClientEventConfiguration* message, if it has not implemented the respective request messages.

If future versions of the RFB protocol provide similar functionality, as defined in these VNC extension messages, a MirrorLink Server and Client shall use these extension messages, unless otherwise stated.

7.2 ByeBye Message

The VNC *ByeBye* message can be sent from both, the VNC Server and VNC Client to inform the other side that the VNC session is going to be terminated intentionally.

The receiving side shall stop sending any further VNC message:

- If the VNC Server wants to terminate the VNC session, it shall send a VNC *ByeBye* message. The VNC Client shall disconnect the TCP connection on reception of that VNC *ByeBye* message.

- If the VNC Client wants to terminate the VNC session, it shall send a VNC *ByeBye* message. The VNC Server shall respond with a VNC *ByeBye* message. The VNC Client shall disconnect the TCP connection on reception of that VNC *ByeBye* message.

The *ByeBye* message is given in Table 6.

Table 6: ByeBye Message

# bytes	Type	Value	Description
1	U8	128	Message-type
1	U8	0	Extension-type
2	U16	0	Payload length

The MirrorLink Server or Client may send a VNC *ByeBye* message in in lieu of a specific response message, which is otherwise mandated from the present document, i.e. replacing the original response message.

The MirrorLink Server or Client may send a VNC *ByeBye* message immediately following a message, which otherwise mandates a response from the receiving device according to the present document. The receiving side then need not respond to the original message, but only to the received VNC *ByeBye* message.

This specifically applies to the following cases:

- The MirrorLink Server's response to a *SetEncoding* message, include the MirrorLink pseudo encoding
- The MirrorLink Client's response to a *ServerDisplayConfiguration* message
- The MirrorLink Server's response to a *ClientDisplayConfiguration* message
- The MirrorLink Client's response to a *ServerEventConfiguration* message
- The MirrorLink Server's response to a *DeviceStatusRequest* message

7.3 Display Configuration Messages

7.3.1 General

The *ServerDisplayConfiguration* and *ClientDisplayConfiguration* message pair exchanges additional display information between the VNC Client and the Server. Based on the received information the VNC Client may change the pixel format, sending a *SetPixelFormat* message. The VNC Server will use this information to optionally provide higher resolution virtual framebuffer copies. The message flow is shown in Figure 5.

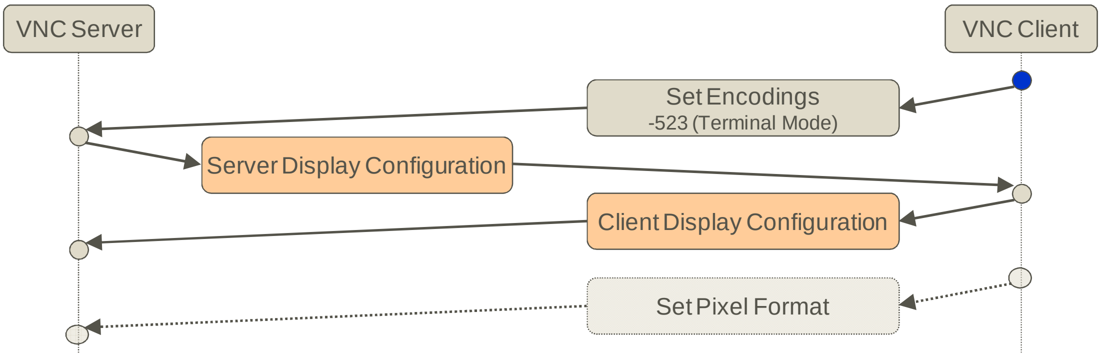

Figure 5: Server and Client Display Configuration

The *ServerDisplayConfiguration* message shall be sent immediately in response to the *SetEncoding* message, indicating MirrorLink (-523) support, prior to any other message. The *ClientDisplayConfiguration* message shall be sent immediately in response to the *ServerDisplayConfiguration* message, prior to any other message.

The *ServerDisplayConfiguration* message is given in Table 7. It will be responded to, by the VNC Client with, a *ClientDisplayConfiguration* message.

Table 7: *ServerDisplayConfiguration* Message

# bytes	Type	Value	Description
1	U8	128	Message-type
1	U8	1	Extension-type
2	U16	12	Payload length
1	U8		MirrorLink Server Major Version
1	U8		MirrorLink Server Minor Version
2	U16	*Bit*	*Framebuffer configuration (1 = yes, 0 = no)*
		[0]	Server-side framebuffer orientation switch available **Deprecated**. Shall be set to "0" (no support).
		[1]	Server-side framebuffer rotation available **Deprecated**. Shall be set to "0" (no support).
		[2]	Server-side framebuffer up-scaling available **Deprecated**. Shall be set to "0" (no support).
		[3]	Server-side framebuffer down-scaling available
		[5]	Server supports *FramebufferAlternativeText* messages. **Deprecated**. Shall be set to "0" (no support).
2	U16		Relative pixel width **Deprecated**. Shall be set to "1".
2	U16		Relative pixel height **Deprecated**. Shall be set to "1".
4	U32	*Bit*	*Pixel format support (1 = yes, 0 = no)*
		[0]	32-bit ARGB 888 (mandatory support for VNC Server)
		[7]	Any other 32-bit format **Deprecated**. Shall be set to "0" (no support).
		[8]	24-bit RGB 888 **Deprecated**. Shall be set to "0" (no support).
		[15]	Any other 24-bit format **Deprecated**. Shall be set to "0" (no support).
		[16]	16-bit RGB 565 (mandatory support for VNC Server)
		[17]	16-bit RGB 555 (15-bit colour depth) **Deprecated**. Shall be set to "0" (no support).
		[18]	16-bit RGB 444 (12-bit colour depth) **Deprecated**. Shall be set to "0" (no support).
		[19]	16-bit RGB 343 (10-bit colour depth) **Deprecated**. Shall be set to "0" (no support).
		[23]	Any other 16-bit format **Deprecated**. Shall be set to "0" (no support).
		[24]	16-bit single colour (grayscale) **Deprecated**. Shall be set to "0" (no support).
		[25]	8-bit single colour (grayscale) **Deprecated**. Shall be set to "0" (no support).

The VNC Server shall support ARGB 888 and RGB 565 colour formats. It may support other formats as well. The VNC Client shall select only colour formats, supported from the VNC Server, using the *SetPixelFormat* message. With respect to the present document, the colour encoding is represented in Table 8 (in case of big endian).

Table 8: Colour Value Parameter (Big Endian)

Parameter	ARGB 888	RGB 565
Bits-per-pixel	32	16
Colour Depth	24	16
Red- max	255	31
Green max	255	63
Blue max	255	31
Red shift	16	11
Green shift	8	5
Blue shift	0	0

In case the bit-per-pixel is larger than the colour depth, the unused bits should be set to Zero.

The *ServerDisplayConfiguration* message should be sent only once. The VNC Client shall ignore all subsequent *ServerDisplayConfiguration* messages.

The *ClientDisplayConfiguration* message is shown in Table 9.

Table 9: *ClientDisplayConfiguration* Message

# bytes	Type	Value	Description
1	U8	128	Message-type
1	U8	2	Extension-type
2	U16	22	Payload length
1	U8		MirrorLink Client Major Version
1	U8		MirrorLink Client Minor Version
2	U16	*Bit*	*Framebuffer Configuration (1 = yes, 0 = no)*
		[0]	Server-side framebuffer orientation switch used **Deprecated**. Shall be set to `"0"` (no support).
		[1]	Server-side framebuffer rotation used **Deprecated**. Shall be set to `"0"` (no support).
		[2]	Client-side framebuffer up-scaling available
		[3]	Client-side framebuffer down-scaling available **Deprecated**. Shall be set to `"0"` (no support).
		[5]	Client supports *FramebufferAlternativeText* messages **Deprecated**. Shall be set to `"0"` (no support).
2	U16		Client display width [pixel]
2	U16		Client display height [pixel]
2	U16		Client display width [mm]
2	U16		Client display height [mm]
2	U16		Distance user - Client display [mm]

# bytes	Type	Value	Description
4	U32		Pixel Format Support **Deprecated**. Shall be set to one of the following values: • `"0x00000001"` (ARGB 888), • `"0x00010000"` (RGB 565) or • `"0x00010001"` (ARGB 888 and RGB 565)
4	U32		Supported resize factors **Deprecated**. Shall be set to `"0x00000000"` (no support).

The MirrorLink Client shall provide correct VNC Client display width and height values in resolution [pixel] and dimension [mm] within the *ClientDisplayConfiguration* message. The given values shall accurately represent the display area on the MirrorLink Client fully dedicated to showing the MirrorLink Server's framebuffer. The resolution [pixel] values shall not be equal to zero (0). The dimension [mm] values shall not be equal to zero (0) except as described in the following cases:

- If either Client display width [mm] or Client display height [mm] is zero (0), the screen area of size shall be recognized as the Reference Display Size at MirrorLink Server side (i.e. 133,3 mm x 80 mm at a distance of 900 mm).

- If the distance is unknown, the value shall be 0, in which case a value of 900 mm shall be used from the MirrorLink Server per default.

The MirrorLink Server shall ignore any Display resolution [pixel] value, where width and/or height are set to zero. If either Client display width [pixel] or Client display height [pixel] is zero (0) within the first *ClientDisplayConfiguration* message, it shall be recognized as the Reference Display Resolution from the MirrorLink Server (i.e. 800 x 480).

The MirrorLink Client shall provide a display, which allows any CCC drive-certified application to be displayed, while the MirrorLink Client is in drive mode. The MirrorLink Client's minimum resolution shall be 800 x 480 on a screen area of size 133,3 mm x 80 mm at a distance of 900 mm. Applications may adapt to the provided screen details, exceeding the minimum resolution, but are not required to do so.

The minimum size shall be adjusted, in case the MirrorLink Client is known to be at a different distance, according to the following formulas:

$$Width_{adjusted} = Width_{at\ 900mm} \cdot \frac{Distance\ [mm]}{900}$$

$$Height_{adjusted} = Height_{at\ 900mm} \cdot \frac{Distance\ [mm]}{900}$$

Implementation Note for MirrorLink 1.0 to 1.2 Clients:

> MirrorLink 1.0 to 1.2 Clients may have a smaller screen size and resolution. This has been within the MirrorLink Client's responsibility. These MirrorLink Clients shall not have a framebuffer resolution smaller than 600 x 400 pixel and shall not have a framebuffer size smaller than 105,75 x 70,5 mm (5" diagonal) at a distance of 900 mm.

The *ClientDisplayConfiguration* message may be sent again, to indicate changes to the VNC Client display configuration. The MirrorLink Server may consider this for changing local scaling or virtual framebuffer setups, sending a *DesktopSizePseudoEncoding* message to the MirrorLink Client.

The MirrorLink major and minor version and the Framebuffer Configuration shall not be changed in any subsequent messages.

The VNC Client should discard any framebuffer content in the *FramebufferUpdate* message upon receiving the Desktop Size Pseudo-rectangle in that *FramebufferUpdate* message and should ensure that the incremental flag is set to zero (false) in the next *FramebufferUpdateRequest* message.

The VNC Client shall provide a valid MirrorLink minor and major version, not higher than the VNC Server supported version. The VNC Server shall use the MirrorLink version provided from the VNC Client.

7.3.2 Framebuffer Scaling (VNC Server)

The MirrorLink VNC Server shall down-scale its framebuffer to fit into the framebuffer resolution provided from the MirrorLink Client in the *ClientDisplayConfiguration* message.

NOTE: A framebuffer resolution fits e.g. within 800 x 480, if its width is equal to or smaller than 800 pixels and its height is equal to or smaller than 480 pixels.

For non-H.264 encoding, the MirrorLink VNC Server shall downscale either to the resolution of the MirrorLink Client or to a fixed resolution, as given in Table 10. The downscaled resolution may have either the horizontal or the vertical resolution smaller than the target resolution, while keeping the other identical to the target resolution. This allows the MirrorLink VNC Server to preserve its framebuffer aspect ratio, as required in Clause 7.3.4.

NOTE: A MirrorLink VNC Server may e.g. downscale from 1 920 x 1 080 pixel to a framebuffer resolution of 800 x 450 pixel.

Table 10: Framebuffer Downscaling Configurations

MirrorLink Client Framebuffer FB$_{\text{Client}}$	MirrorLink Server Framebuffer FB$_{\text{Server}}$	
	FB$_{\text{Server}}$ < 1 024 x 600	FB$_{\text{Server}}$ ≥ 1 024 x 600
FB$_{\text{Client}}$ < 800 x 480 (MirrorLink 1.0 - 1.2 Clients only)	FB$_{\text{Client}}$ or 800 x 480	FB$_{\text{Client}}$ or 800 x 480
FB$_{\text{Client}}$ < 1 024 x 600	FB$_{\text{Client}}$ or 800 x 480	FB$_{\text{Client}}$ or 800 x 480
FB$_{\text{Client}}$ ≥ 1 024 x 600	N/A	FB$_{\text{Client}}$, 1 024 x 600 or 800 x 480

For H.264 encoding, the MirrorLink VNC Server shall downscale either to the MirrorLink Client's framebuffer resolution or to the closest supported framebuffer resolution, which is still within the MirrorLink Client's framebuffer resolution, as included in the *ClientDisplayConfiguration* message. This allows the MirrorLink VNC Server to better optimize H.264 encoding. Nevertheless, the MirrorLink VNC Server shall not downscale below the Reference Screen resolution of 800 x 480 pixel, adjusted for the Server's framebuffer aspect ratio.

The MirrorLink VNC Server shall not scale, if the VNC Client display width or height [pixel] within the *ClientDisplayConfiguration* message is unknown.

If required, the MirrorLink VNC Server shall down-scale on receiving a *ClientDisplayConfiguration* message and the MirrorLink VNC Server shall indicate that it started to downscale in the 1st *FramebufferUpdate* message using Desktop Size Pseudo Encoding rectangle; framebuffer data shall not be included.

Implementation Note for MirrorLink 1.0 to 1.2 Servers:

> MirrorLink 1.0 to 1.2 Servers may include framebuffer and context information data in framebuffer updates, containing a Desktop Size Pseudo Encoding rectangle. This framebuffer data shall be ignored from MirrorLink 1.3 Clients.

The message flow the VNC Server shall follow, is shown in Figure 6.

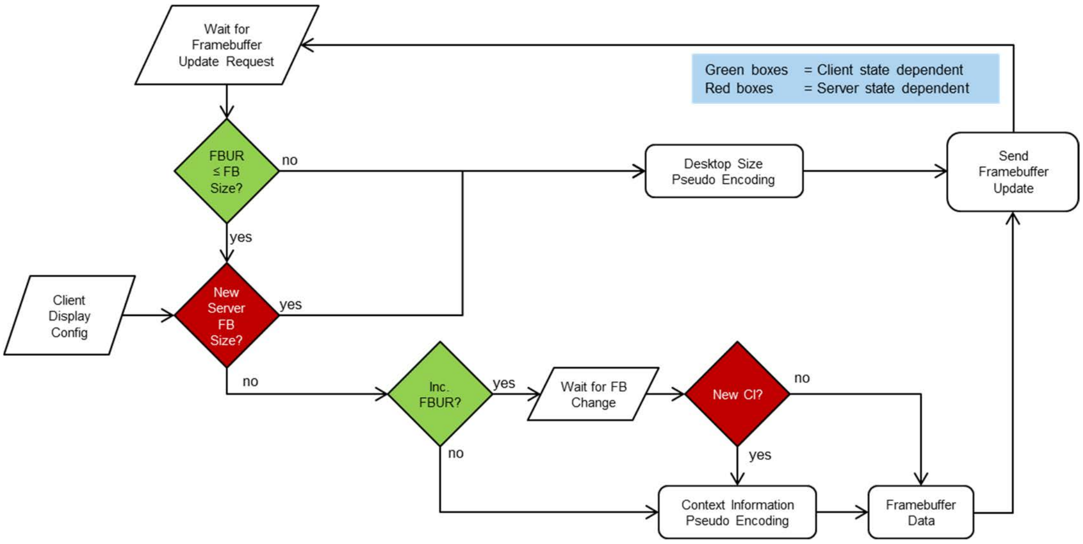
Wait for Framebuffer Update Request
Green boxes = Client state dependent
Red boxes = Server state dependent
FBUR ≤ FB Size?
no
yes
Desktop Size Pseudo Encoding
Send Framebuffer Update
Client Display Config
New Server FB Size?
yes
no
Inc. FBUR?
yes
Wait for FB Change
New CI?
no
no
yes
Context Information Pseudo Encoding
Framebuffer Data

Figure 6: Framebuffer Scaling (VNC Server) - Message Flow

The MirrorLink Server shall preserve the framebuffer's aspect ratio during scaling.

The MirrorLink Server shall not up-scale.

7.3.3 Framebuffer Scaling (VNC Client)

The MirrorLink Client shall support up-scaling to the framebuffer resolution provided in *ClientDisplayConfiguration* message.

The MirrorLink Client shall provide valid VNC Client display width and height [pixel] within the *ClientDisplayConfiguration* message.

The MirrorLink Client may request a smaller framebuffer resolution from the MirrorLink Server (sending a *ClientDisplayConfiguration* message), and then perform internal up-scaling, if the requested resolution equals or exceeds 800 x 480. This is transparent to the MirrorLink Server.

The MirrorLink Client may add pad rows or columns, but shall not add more than 7 pad-pixel rows and more than 7 pad-pixel columns at the same time. Therefore, the MirrorLink Client shall scale the incoming MirrorLink Server framebuffer to match at least one dimension of its reported framebuffer resolutions.

7.3.4 Handling of Different Framebuffer Aspect Ratios

The Framebuffer Aspect Ratio (FAR) is defined as the horizontal display resolution [pixel] divided by the vertical display resolution [pixel]. The FAR is also referred to as the storage aspect ratio, as those pixels are stored in memory. The MirrorLink Server and MirrorLink Client displays can have different Framebuffer Aspect Ratios.

The MirrorLink Server shall not stretch its framebuffer to compensate for any difference in the Framebuffer Aspect Ratio, but it may add pad rows or columns to the framebuffer, prior to transmitting it. Padding shall not exceed the MirrorLink Client's framebuffer resolution, as given in the *ClientDisplayConfiguration* message. The MirrorLink Server shall use the padded resolution within any VNC *ServerInit* and *DesktopSizePseudoEncoding* message.

EXAMPLE 1: A MirrorLink Server with a Display resolution of 1 280 x 720 pixel (FAR = 16:9) is providing its framebuffer to a MirrorLink Client with a Display Resolution of 800 x 480 (FAR = 5:3). The MirrorLink Server will need to downscale its framebuffer to 800 x 450 to be within the Client framebuffer resolution. The Server may add 30 rows to its framebuffer to match the Client's FAR.

EXAMPLE 2: A MirrorLink Server with a Display resolution of 1 280 x 720 pixel (FAR = 16:9) is providing its framebuffer to a MirrorLink Client with a Display Resolution of 1 024 x 576 (FAR = 16:9). The MirrorLink Server will need to downscale its framebuffer to 1 024 x 576 to be within the Client framebuffer resolution. The Server shall not add any pad rows or columns to its framebuffer.

The VNC Server shall use a framebuffer resolution in the VNC *ServerInit* and *DesktopSizePseudoEncoding* message, which results in a framebuffer aspect ratio between 1 and 1,77 (16:9) in Classic MirrorLink Mode and between 1 and 1.90 in Immersive Mode. The aspect ratio refers to the framebuffer, which is send to the MirrorLink Client. For MirrorLink Servers, using a virtual framebuffer, this aspect ratio may be different from the aspect ratio of the MirrorLink Server's physical screen. The higher maximum allowed aspect ratio in Immersive Mode results from the additionally required 1 cm in horizontal screen size, as specified in [6].

Implementation Note for MirrorLink 1.0 to 1.2 Servers:

MirrorLink 1.0 to 1.2 Servers may have an extended framebuffer aspect ratio of up to 2.0.

7.3.5 Handling of Server Pixel Aspect Ratios

The Pixel Aspect Ratio (PAR) is defined as the width of a 1 (one) pixel divided by the height of 1 (one) pixel. If the PAR of the MirrorLink Server Display does not match the PAR of the MirrorLink Client display, the MirrorLink Server's display content will look distorted on the MirrorLink Client's display (e.g. a square will not be a square and a circle will not be a circle).

NOTE: A PAR of 1 is common with all modern Smartphone displays. A PAR of not equal to 1 has been used in TV sets.

The MirrorLink Server shall compensate for a Server PAR not equal to 1 (one), prior to transmitting its framebuffer to the MirrorLink Client. The MirrorLink Server shall use the PAR-compensated resolution within any VNC *ServerInit* and *DesktopSizePseudoEncoding* message. The MirrorLink Server shall set the relative pixel width and height values in the *ServerDisplayConfiguration* message to 1.

EXAMPLE 1: A MirrorLink Server with a Display resolution of 1 280 x 720 pixel and a Display size of 4:3, has a pixel aspect ratio of (4/1 280) / (3/720) = 3/4. The Server will need to re-scale its framebuffer to 960 x 720 to achieve a PAR of 1.

EXAMPLE 2: A MirrorLink Server with a Display resolution of 1 280 x 720 pixel and a Display size of 16:9, has a pixel aspect ratio of (16/1 280) / (9/720) = 1. The Server will not need to re-scale its framebuffer.

The MirrorLink Client should compensate for a Client PAR not equal to 1 (one). If the MirrorLink Client compensates for a Client PAR not equal to 1 (one), it shall adjust the framebuffer resolution within the *ClientDisplayConfiguration* message to fully match the PAR compensation (i.e. the framebuffer resolution describes squared pixels). The reported physical dimensions shall not be adjusted in this case.

7.3.6 Handling of Application, Framebuffer and Display Orientation

The application's user interface (UI) can be in Landscape or Portrait orientation. The MirrorLink Server will always send a landscape-oriented framebuffer to the MirrorLink Client, which will display it on its display in a landscape-oriented view as well. To embed a portrait-oriented application UI into a landscape-oriented (virtual) framebuffer, the MirrorLink Server will need to down-scale the UI and add side-bars, prior sending it to the MirrorLink Client, as shown in Figure 7.

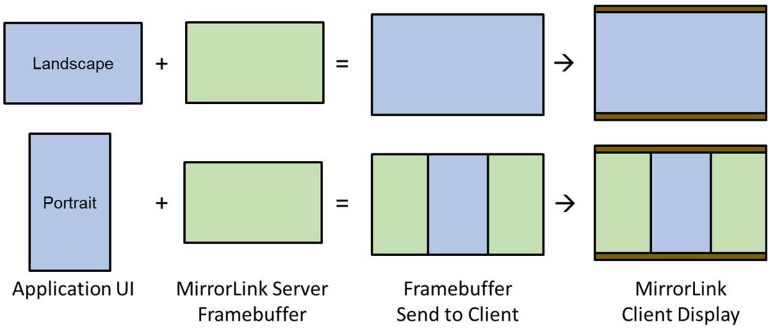
Landscape
+
=
→
Portrait
+
=
→
Application UI
MirrorLink Server Framebuffer
Framebuffer Send to Client
MirrorLink Client Display

Figure 7: Handling of Application UI Orientation

The MirrorLink Server shall launch any base or drive-certified application in Landscape orientation and shall lock the application into that orientation.

In case the MirrorLink Server launches a non-certified application, it should launch the application in Landscape orientation and lock the application into that orientation. In case the non-certified application requests Portrait orientation, the MirrorLink Server shall embed the Portrait oriented application UI into the Landscape oriented framebuffer send to the MirrorLink Client, as shown in Figure 7, or shall treat the application as being blocked from the MirrorLink Client, as specified in Clause 7.8. The MirrorLink Server should centre the portrait-oriented application UI within the framebuffer sent to the Client. The colour of any added side bars should match the MirrorLink Server's default background colour. A MirrorLink Client cannot distinguish an embedded Portrait orient application UI from a regular landscape UI. This mode of operation is only available to MirrorLink Clients that support non-certified applications in Park Mode.

The MirrorLink Server shall not rotate the framebuffer.

7.4 Event Configuration Messages

The *ServerEventConfiguration* and *ClientEventConfiguration* message pair provides additional information about event handling, i.e. which key and pointer events are natively supported on the VNC Server and Client. This information helps the Server to map specific incoming VNC Client events to Server events.

The message flow is shown in Figure 8.

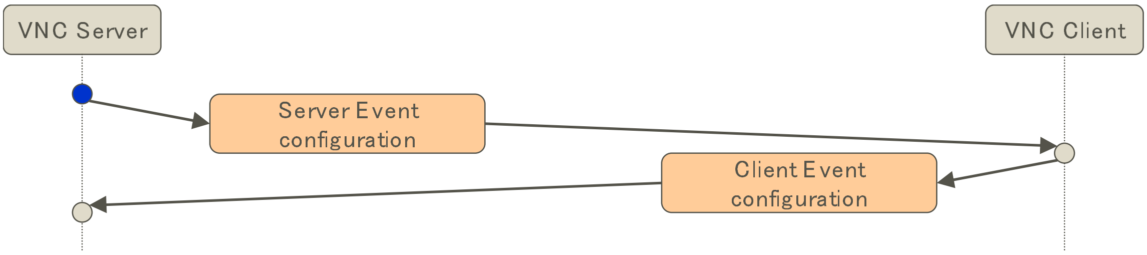

Figure 8: Server and Client Event Configuration

The *ServerEventConfiguration* and *ClientEventConfiguration* messages have to follow the rules, given next:

- The MirrorLink Server shall provide all key events, it has mapped to platform specific events, as detailed in the respective platform specific specifications. Non-mapped events shall be set to "0".
- The MirrorLink Client shall provide all key events it natively supports. This may include soft buttons, e.g. rendered on a touch screen.
- The MirrorLink Client shall be able to utilize all key events, advertised in the *ClientEventConfiguration* message.
- The MirrorLink Server shall not assume that the MirrorLink Client will support all key events, supported from the MirrorLink Server.

The *ServerEventConfiguration* message shall be sent immediately in response to the *SetEncoding* message, indicating MirrorLink (-523) support, but after the *ServerDisplayConfiguration* has been sent. The message may be delayed only until reception of the *ClientDisplayConfiguration* message. The *ClientEventConfiguration* message shall be sent immediately in response to the *ServerEventConfiguration* message, prior any other message.

The *ServerEventConfiguration* message is given in Table 11.

Table 11: *ServerEventConfiguration* Message

# bytes	Type	Value	Description
1	U8	128	Message-type
1	U8	3	Extension-type
2	U16	28	Payload length
2	U16		Keyboard layout - Language code (according ISO 639-1 [9])
2	U16		Keyboard layout - Country code (according ISO 3166-1 [10] alpha-2)

# bytes	Type	Value	Description
2	U16		UI Language - Language code (according ISO 639-1 [9])
2	U16		UI Language - Country code (according ISO 3166-1 [10] alpha-2)
4	U32	*Bit l*	*Knob keys* (Bit mask according Table A.1) • '1': Server supports knob key events • '0': Server does not support knob key events
4	U32	*Bit m*	*Device keys* (Bit mask according Table B.2) • '1': Server supports device key events • '0': Server does not support the key events
4	U32	*Bit n*	*Multimedia keys* (Bit mask according Table B.2) • '1': Server supports multimedia key events • '0': Server does not support the multimedia key events
4	U32	*Bit*	*Key related (1 = support, 0 = no support)*
		[0]	ITU keypad events **Deprecated**. Shall be set to `"0"` (no support).
		[1]	Virtual keyboard trigger support **Deprecated**. Shall be set to `"0"` (no support).
		[2]	Key event listing support **Deprecated**. Shall be set to `"0"` (no support).
		[3]	Event mapping support (Shall be '1')
		[15:8]	Number of Function keys, the VNC Server supports • Key events start with Function_Key 0, no subsequent gaps
4	U32	*Bit*	*Pointer related (1 = support, 0 = no support)*
		[0]	Pointer events
		[1]	Touch events
		[15:8]	Pointer event button mask (according RFB spec) Shall be 0x00 if the VNC Server does not support pointer events.
		[23:16]	(Number of supported simultaneous events minus one) within a touch event, e.g. a value of 2 indicates a support of 3 parallel events. Shall be 0x00, if the VNC Server does not support touch events.
		[31:24]	Touch event pressure mask, supported from the VNC Server. Shall be 0x00, if the VNC Server does not support touch events.

If the VNC Server supports touch events, the pressure mask indicates how many pressure levels can be distinguished at the VNC Server:

- A pressure mask of 0x01 indicates that only two states can be distinguished in a touch event, based on the pressure levels, one release and one press state.

- A pressure mask of 0xFF indicates that 256 touch events can be distinguished in a touch event, based on the pressure levels, one release and 255 different press states.

- A pressure mask of 0x00 shall not be used if the "Touch events" bit is enabled.

The pressure mask shall be continuously filled with '1's, starting from bit 24.

The payload structure of the *ClientEventConfiguration* message is fully symmetrical with the *ServerEventConfiguration* message, as shown in Table 12.

Table 12: *ClientEventConfiguration* Message

# bytes	Type	Value	Description
1	U8	128	Message-type
1	U8	4	Extension-type
2	U16	28	Payload length
2	U16		Keyboard layout - Language code (according ISO 639-1 [9])

# bytes	Type	Value	Description
2	U16		Keyboard layout - Country code (according ISO 3166-1 [10] alpha-2)
2	U16		UI Language - Language code (according ISO 639-1 [9])
2	U16		UI Language - Country code (according ISO 3166-1 [10] alpha-2)
4	U32	*Bit l*	*Knob keys* (Bit mask according Table A.1) • '1': Client supports knob key events • '0': Client does not support knob key events
4	U32	*Bit m*	*Device keys* (Bit mask according Table B.2) • '1': Client supports device key events • '0': Client does not support device key events
4	U32	*Bit n*	*Multimedia keys* (Bit mask according Table B.2) • '1': Client supports multimedia key events • '0': Client does not support multimedia key events
4	U32	*Bit*	*Key related (1 = support, 0 = no support)*
		[0]	ITU keypad events **Deprecated**. Shall be set to "0" (no support).
		[1]	Virtual keyboard trigger support **Deprecated**. Shall be set to "0" (no support).
		[2]	Key event listing support **Deprecated**. Shall be set to "0" (no support).
		[3]	Event Mapping support **Deprecated**. Shall be set to "0" (no support).
		[15:8]	Number of Function keys, the VNC Client supports • Key events start with Function_Key 0, no subsequent gaps
4	U32	*Bit*	*Pointer related (1 = support, 0 = no support)*
		[0]	Pointer events
		[1]	Touch events
		[15:8]	Pointer event button mask (according RFB spec) Shall be 0x00, if the VNC Client does not support pointer events.
		[23 :16]	(Number of supported simultaneous touch events -1) within a touch event, e.g. a value of 2 indicates a support of 3 parallel events. Shall be 0x00, if the VNC Client does not support touch events.
		[31:24]	Touch event pressure mask, supported from the VNC Client. Shall be 0x00, if the VNC Client does not support touch events.

A MirrorLink Client, not supporting Pointer events triggered from the touch enabled MirrorLink Client screen, shall support the Knob_2D_0_shift_push event and the knob event pairs listed below and shall enable them in the *ClientEventConfiguration* message:

- Knob_2D_0_shift_right and Knob_2D_0_shift_left
- Knob_2D_0_shift_up and Knob_2D_0_shift_down
- Knob_2D_0_rotate_z and Knob_2D_0_rotate_Z
- Device_Backward

A MirrorLink Client may support directional events, e.g. horizontal left/right and/or vertical up/down events via the steering wheel, in addition to touch events. In case these are made available to control MirrorLink applications, the MirrorLink Client shall advertise them to the MirrorLink Server as a knob 0 within the *ClientEventConfiguration* message. Support for an example horizontal & vertical steering wheel joystick is shown below:

- Bit 0: Shift along x axis enabled
- Bit 1: Shift along y axis enabled
- Bit 2: Shift along xy diagonals disabled

- Bit 3: Push along z axis disabled
- Bit 4: Pull along z axis disabled
- Bit 5: Rotation around x axis disabled
- Bit 6: Rotation around y axis disabled
- Bit 7: Rotation around z axis disabled

A MirrorLink Client should make directional and other MirrorLink relevant control events (like phone call accept/reject), implemented on the steering wheel, available to MirrorLink applications. This can significantly improve usability of applications.

Implementation Note:

> In case a MirrorLink Client supports more than 1 rotary knob controller or directional events, the MirrorLink Client should treat them as one rotary knob controller, if the MirrorLink Server advertise support for only 1 rotary knob. Adding these as a second knob is allowed, but applications will not be able to receive those.

A MirrorLink Server shall pass the following knob events to the applications and shall enable them in the *ServerEventConfiguration* message:

- Knob_2D_0_shift_push event
- Knob_2D_0_shift_right, Knob_2D_0_shift_left
- Knob_2D_0_shift_up, Knob_2D_0_shift_down
- Knob_2D_0_rotate_z, Knob_2D_0_rotate_Z
- Device_Backward

A MirrorLink Server shall not remap the above listed knob events to different key event values, other than to allow MirrorLink applications to navigate and select user input elements.

A MirrorLink Server shall pass all other received MirrorLink events to MirrorLink-Certified and Member-Certified Applications. The MirrorLink Server may remap them to other key event values.

If the VNC Client supports touch events, the pressure mask indicates how many pressure levels can be distinguished at the VNC Client. It shall follow the same ruling as the VNC Server side pressure mask.

A MirrorLink Server shall support Pointer events, at least to navigate and select user input elements, and shall enable it in the *ServerEventConfiguration* message.

The keyboard layout and UI language setting, within the *Server* and *Client Event Configuration* message reflect the respective current settings of the MirrorLink Server and Client. The MirrorLink Client should follow the language settings of the MirrorLink Server.

Device_Backward Event

> The *Device_Backward* event allows the consumer to navigate within application. The behaviour is platform specific. Requirements for supporting the *Device_Backward* event are defined in [6].
>
> A MirrorLink Client, supporting a *Device_Backward* event, shall enable the *Device_Backward* event flag in the *Client Event Configuration* message.
>
> The handling of the *Device_Backward* event on the MirrorLink Server shall not lead to uncertified content during drive mode.

NOTE: To pre-emptively prevent framebuffer blocking, the MirrorLink Server can either switch to the MirrorLink Server's previously launched (drive-certified) home screen application or request the MirrorLink Client to switch to its native UI, using `"Switch to MirrorLink Client native UI"` (0xF000FFFF) context information, as defined in Clause 8.3.

In case the *Device_Backward* key event is nevertheless leading to uncertified content, the MirrorLink Client shall send framebuffer blocking notification, on which the MirrorLink Server shall immediately respond, as specified in Clause 7.8.

Implementation Note for MirrorLink 1.0 to 1.2 Clients:

MirrorLink 1.0 to 1.2 Clients may not support a *Device_Backward* event, as required.

Device_Home Event

The *Device_Home* event allows the consumer to switch back to a start screen, the consumer is familiar with, either on the MirrorLink Server's Home Screen application or on the MirrorLink Client. Requirements for supporting the *Device_Home* event are defined in [6].

A MirrorLink Client, supporting a *Device_Home* event, shall enable the *Device_Home* event flag in the *Client Event Configuration* message.

The handling of the *Device_Home* event on the MirrorLink Server shall not lead to uncertified content during drive mode.

NOTE: To pre-emptively prevent framebuffer blocking, the MirrorLink Server can either switch to the MirrorLink Server's (drive-certified) home screen application or request the MirrorLink Client to switch to its native UI, using `"Switch to MirrorLink Client native UI"` (0xF000FFFF) context information, as defined in Clause 8.3.

In case the *Device_Home* key event is nevertheless leading to uncertified content, the MirrorLink Client shall send framebuffer blocking notification, on which the MirrorLink Server shall immediately respond, as specified in Clause 7.8.

Implementation Note for MirrorLink 1.0 to 1.2 Clients:

MirrorLink 1.0 to 1.2 Client and Server devices may not support a *Device_Home* event.

7.5 Event Mapping Messages

The *EventMapping* and *EventMappingRequest* message pair provide MirrorLink 1.0 to 1.2 VNC Clients with information about the VNC Server mapping of high key symbol values, as listed in Annex B. The VNC Client can send an *EventMappingRequest* message at any time, requesting a specific mapping within the VNC Server. The VNC Server shall always send an *EventMapping* message in response to an *EventMappingRequest* message. The VNC Server shall not change any event mapping locally.

Event Mapping has been deprecated in MirrorLink 1.3. MirrorLink 1.3 Client shall not send any *EventMappingRequest* message.

The message flow follows the steps, as shown in Figure 9.

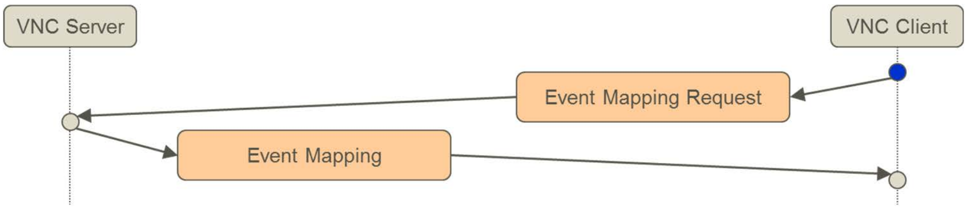

Figure 9: Example *EventMapping* Message Flow

The MirrorLink Server shall respond to any *EventMappingRequest* message immediately.

The Event Mapping message is given in Table 13.

Table 13: *EventMapping* Message

# bytes	Type	Value	Description
1	U8	128	Message-type
1	U8	5	Extension-type
2	U16	8	Payload length
4	U32		Client Key Symbol Value
4	U32		Server Key Symbol Value - Set to 0, Client key value is not mapped - Set to Client Key value, if Client key is mapped

The Default Mapping Request message is given in Table 14.

Table 14: *EventMappingRequest* Message

# bytes	Type	Value	Description
1	U8	128	Message-type
1	U8	6	Extension-type
2	U16	8	Payload length
4	U32		Client Key Symbol Value
4	U32		Server Key Symbol Value **Deprecated**. Shall be ignored.

A MirrorLink Server shall not send an *EventMapping* message unsolicited, without a prior *EventMapingRequest* message.

7.6 Device Status Messages

The *DeviceStatus* and *DeviceStatusRequest* message pair provides the VNC Client with status information of specific device settings at the VNC Server and the ability to set them. The VNC Server shall inform the VNC Client about any status change. The message flow is shown in Figure 10.

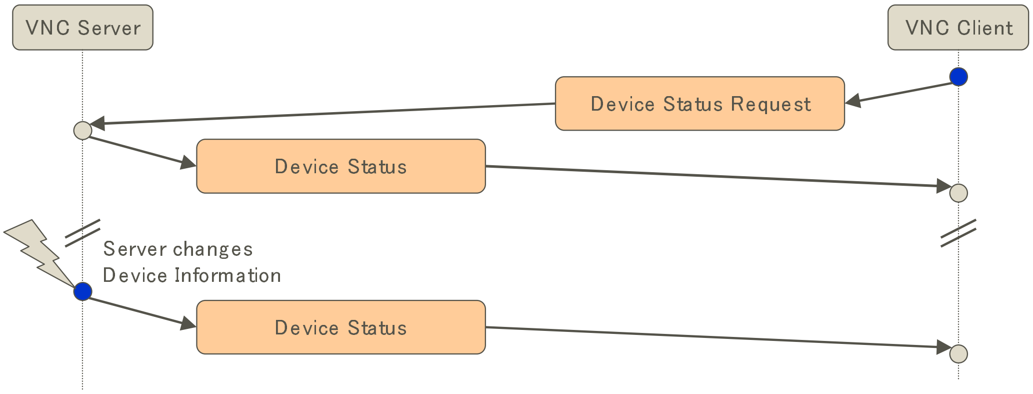

Figure 10: Example *DeviceStatusMessage* Flow

The VNC Server shall immediately respond to a *DeviceStatusRequest*, after the action has been completed on the MirrorLink Server, which shall not take longer than 1 s.

The Device Status message is given in Table 15.

Table 15: *DeviceStatus* Message

# bytes	Type	Value	Description
1	U8	128	Message-type
1	U8	11	Extension-type
2	U16	4	Payload length
4	U32	*Bit*	*Status of Device Features* *(00 = unknown, 01 = reserved,* *10 = disabled, 11 = enabled)*
		[1:0]	Key-lock **Deprecated:** Shall be set to `"00"` (unknown)
		[3:2]	Device-lock (In device-lock state, the MirrorLink Server is locked and need to respond to remote key and pointer events. The User may need to enter a PIN code to un-lock the device; PIN code entry need not be possible via the MirrorLink Client
		[5:4]	Screen saver **Deprecated:** Shall be set to `"00"` (unknown)
		[7:6]	Night mode
		[9:8]	Voice control input on MirrorLink Server The MirrorLink Server may use these bits to indicate to the MirrorLink Client, that it expects voice control input to be enabled or disabled.
		[11:10]	Microphone input on MirrorLink Client (Microphone input on MirrorLink Client; Enables or disables the microphone input at the MirrorLink Client). The audio from the microphone should be treated as conversational audio, if Voice Control Input status is not enabled. The audio from the microphone shall be treated as voice command audio, if Voice Control Input status is enabled. The MirrorLink Server may use these bits to indicate to the MirrorLink Client, that audio input is expected from the microphone (e.g. VoIP call). These bits shall not be used to indicate end or start of phone call for BT HFP case.
		[17:16]	Driver Distraction Avoidance (00 = unknown: Shall not use 01 = reserved: Shall not use 10 = disabled: MirrorLink Client is in non-restricted driving mode 11 = enabled: MirrorLink Client is in restricted driving mode)
		[26:24]	Absolute Framebuffer rotation (clock-wise) **Deprecated**. Shall be set to `"100"` (0º)
		[28:27]	Framebuffer orientation **Deprecated:** Shall be set to `"10"` (Landscape)

The *DeviceStatusRequest* message is given in Table 16.

Table 16: *DeviceStatusRequest* Message

# bytes	Type	Value	Description
1	U8	128	Message-type
1	U8	12	Extension-type
2	U16	4	Payload length

# bytes	Type	Value	Description
4	U32	*Bit*	*Status of Device Features* *(00 = ignore, 01 =* reserved *10 = disable, 11 = enable)*
		[1:0]	Key-lock **Deprecated:** Shall be set to `"00"` (ignore)
		[3:2]	Device lock **Deprecated:** Shall be set to `"00"` (ignore)
		[5:4]	Screen saver **Deprecated:** Shall be set to `"00"` (ignore)
		[7:6]	Night mode (run application in night mode; device status flag provided via the MirrorLink API)
		[9:8]	Voice input (route the incoming audio stream to a voice recognition engine on the mobile device) The MirrorLink Client shall use this flag only, if the voice command is streamed via RTP. In case an existing BT HFP connection is used and Voice Recognition Activation is supported by both Hands-Free unit and Audio Gateway, the MirrorLink Client shall use the BT HFP voice activation mechanism (AT + BVRA command as specified in [2]) instead.
		[11:10]	Microphone input on MirrorLink Client routed from microphone to the MirrorLink Server
		[17:16]	Driver Distraction Avoidance (MirrorLink Client is in restricted driving mode (enabled), or non-restricted driving mode (disabled))
		[26:24]	Absolute Framebuffer rotation (clock-wise) **Deprecated**. Shall be set to `"000"` (ignore) or `"100"` (0º)
		[28:27]	Framebuffer orientation **Deprecated:** Shall be set to `"10"` (Landscape)

The VNC Server may ignore some or all device features as set in the *DeviceStatusRequest* message, if not specified otherwise below. The VNC Client should be able to detect this from the received *DeviceStatus* message, following the original *DeviceStatusRequest* message.

The VNC Client may set a flag to `"ignore"` (unless specified otherwise), telling the VNC Server that it does not expect the VNC Server to react to it. The VNC Server shall either report its current setting or report `"unknown"`.

The VNC Client should not use the `"reserved"` value in any of the *DeviceStatusRequest* message features. If the VNC Server receives a Device Status feature with the `"reserved"` value, it should consider the value being `"ignore"`.

The VNC Server should not use the `"reserved"` value in any of the *DeviceStatus* message features. If the VNC Client receives a *DeviceStatus* feature with the `"reserved"` value, it should consider the value being `"unknown"`.

The MirrorLink Client should send *DeviceStatusRequest* message only, when the MirrorLink Client want the MirrorLink Server to change the current status. The MirrorLink Client should set `"ignore"` for status items, which MirrorLink Client does not want the MirrorLink Server to change.

Driver Distraction Avoidance

The VNC Client shall send a *DeviceStatusRequest* message with the Driver Distraction Avoidance flag set appropriately, if the MirrorLink Client display, which is used for displaying the VNC content stream, becomes subject to driver distraction avoidance, or whenever the driving mode changes. The Driving Mode defines the state, whether the MirrorLink Client can display application user interfaces in a restricted or non-restricted mode. The Driving Mode is used within Application Certification, as specified in the UPnP Application Server Service's *A_ARG_TYPE_AppCertificateInfo* state variable (for details see [4]).

The MirrorLink Server and Client shall only use `"enabled"/"disabled"` values for the Driver Distraction Avoidance flag. The MirrorLink Server shall not change the Driver Distraction Avoidance status on its own. The MirrorLink Server shall follow the requests from the MirrorLink Client regarding this flag.

Framebuffer Orientation

The MirrorLink Server shall prevent automatic orientation change (into Portrait mode) and shall deny any application request to switch to Portrait mode. The framebuffer shall be locked into Landscape orientation from the MirrorLink Server.

Rotation

The MirrorLink Server shall prevent automatic flipping (180-degree rotation), due to accelerometer or other sensor readings. The framebuffer shall not be rotated from the MirrorLink Server.

Voice Input & Microphone Input

The MirrorLink Server and Client shall correctly follow the Device Status flags for Voice Input, and Mic Input, if they support the underlying use cases; in that case they shall only use `"enabled"/"disabled"` values for them.

The MirrorLink Client shall respond with `"disabled"` if the Mic Input is currently occupied internally, in response to the MirrorLink Server enabling the Mic Input, as shown in Figure 8 of [3].

The MirrorLink Server and MirrorLink Client shall only use `"enable"` or `"disable"` values, if the MirrorLink Client is within a Voice Control or Phone Call over RTP session, or intending to start or end it; otherwise the MirrorLink Client and MirrorLink Server shall use the `"ignore"` or `"unknown"` values.

Device Lock

The MirrorLink Server shall not automatically enable device lock, while being in a MirrorLink session. The consumer may manually enable the device lock from the MirrorLink Server device (e.g. pressing a power button). Dependent on the supported functionality, the following behavior is defined.

NOTE: A VNC Session is considered active if and only if the MirrorLink Server can actively respond to *FramebufferUpdateRequest* messages.

MirrorLink Client Notification (VNC Session cannot be kept active)

In case the MirrorLink Server cannot keep the ongoing VNC session active, the MirrorLink Client is responsible for providing a notification to the consumer. The MirrorLink Server shall send a *DeviceStatus* message, with Device Lock enabled, prior going into the Device Lock or when it is in Device Lock at the begin of a VNC session. The MirrorLink Client shall inform the consumer when the MirrorLink Server goes into device lock during a MirrorLink session or when the Server is already in Device Lock at the beginning of a VNC session.

Implementation Note:

A MirrorLink 1.1 Server may not send a Device Lock message, when the Server is already in Device Lock at the beginning of a VNC session. No notification needs to be provided in this case.

MirrorLink Server Notification (VNC Session stays active)

In case the MirrorLink Server can keep the VNC session active, it shall present the *Device Lock* notification on the MirrorLink Client display, i.e. the MirrorLink Client is not aware of the MirrorLink Server being in *Device Lock*, as the remote framebuffer session will stay active and responsive, and no error codes are provided from the UPnP layer.

The MirrorLink Server shall not send a Device Status message, with Device Lock enabled. The MirrorLink Server shall present a Device Lock notification via its remoted framebuffer.

The presented Device Lock notification should give the consumer the option to remotely unlock the device. In case of a password protected device lock, remote unlock may be impossible.

The notification shall belong to a CCC or Member drive-certified application for the connected MirrorLink Client. The notification shall be shown in Landscape orientation. The MirrorLink Server shall identify the *Device Lock* notification screen using the *appCategory* value of `"Device Unlock Notification"` within the Context Information messaging; preferable as the first entry in the Context Information.

Use of UPnP Actions is Restricted

In case the MirrorLink Server restricts the use of some (or all) UPnP actions, it shall send UPnP responses with the Error Code 815 (`"Device Locked"`) or it shall send *SSDP::byebye* messages. The consumer may not be able to remotely disable the device lock on the MirrorLink Server via the MirrorLink Client.

7.7 Content Attestation Messages

The *ContentAttestationRequest* and *ContentAttestationResponse* message pair allows the MirrorLink Client to securely verify the content stream received from the MirrorLink Server. This message pair allows the MirrorLink Client to detect the following security risks originating from a man-in-the-middle attack:

1) Modification of framebuffer content, with unwanted content

2) Modification of context information, with more favourable settings

To address those risks, the VNC Client can challenge, at any time, the VNC Server to provide a signed response, containing framebuffer characteristics, allowing them to identify and to minimize the risks.

The Contest Attestation message flow is shown in Figure 11.

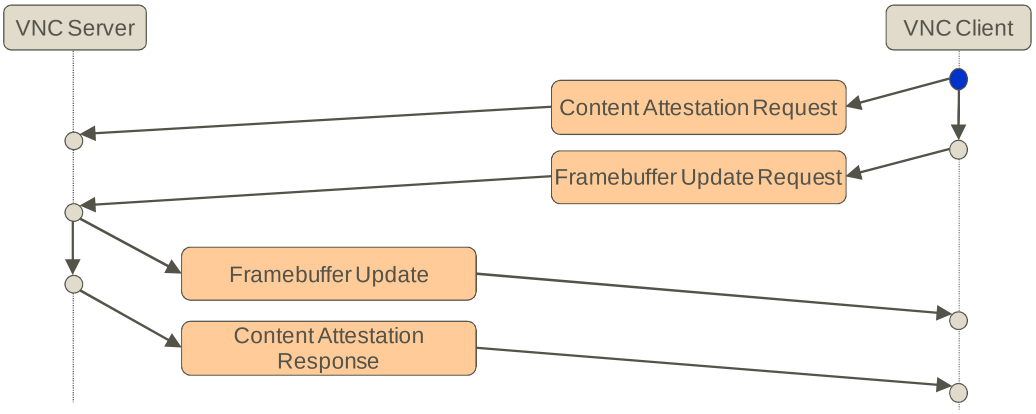
VNC Server
VNC Client
Content Attestation Request
Framebuffer Update Request
Framebuffer Update
Content Attestation Response

Figure 11: Example Content Attestation Message Flow

If the MirrorLink Server provides an *applicationPublicKey* bound to the VNC Server during a DAP session, the MirrorLink Server shall support VNC Content attestation, providing a *ContentAttestationResponse* messages in response to a *ContentAttestationRequest* message.

The VNC Server shall immediately respond to a *ContentAttestationRequest*, after sending the next *FramebufferUpdate*. The VNC Server shall include full context information to the *FramebufferUpdate* message, even if the context has not changed.

The VNC Client shall send the *ContentAttestationRequest* message right before the next *FramebufferUpdateRequest* message. This is required as the MirrorLink Server shall provide the *ContentAttestationResponse* together with the *FramebufferUpdate* message.

The *ContentAttestationResponse* message is given in Table 17.

Table 17: *ContentAttestationResponse* Message

# bytes	Type	Value	Description
1	U8	128	Message-type
1	U8	13	Extension-type
2	U16	122 (Maximum size)	Payload length
2	U16	*Bit*	*SignedInfo Flag* *Defines, what has been attested and included into the hash ('1' = include, '0' = do not include)* NOTE: The MirrorLink Server may choose to attest different content than what was requested by the Client, i.e. the SignedInfo flag set in Content Attestation Response may be different from the one in Content Attestation Request. It is up to the MirrorLink Client to decide whether such attestation is acceptable.
		[0]	SignedInfo includes context information pseudo encoding as provided within the last framebuffer update NOTE: Signature is calculated over all context information pseudo encoding rectangles (as defined in Table 26) in the *FramebufferUpdate* message, i.e. excluding the 4-byte message header and any regular framebuffer encoding.
		[1]	SignedInfo includes the framebuffer content, as provided with the last framebuffer update NOTE: Signature is calculated over all rectangles in the *FramebufferUpdate* message, i.e. excluding the 4-byte message header and any pseudo encoded rectangle (e.g. context information or desktop size).
		[2]	SignedInfo includes number of updated framebuffer pixels sent since previous *ContentAttestationResponse* message NOTE: Signature is calculated over $N = \sum_j \sum_i width_i \cdot height_i$, where i identifies the different rectangles within the regular framebuffer update j, i.e. excluding any pseudo encoded rectangle. N is a 32-bit unsigned integer (in network byte-order).
2	U16	*Value*	*Error code*
		0	Success - no change to SignedInfo flag
		1	Success - with change to SignedInfo flag
		2	Success - no signature added, no change to SignedInfo flag
		3	Success - no signature added, with change to SignedInfo flag
		128	Error - no session key
		129	Error - content attestation not implemented
		255	Error - other error

# bytes	Type	Value	Description
18 - 86	Array of U8		SignedInfo (Size dependent of SignedInfo Flag)
32	Array of U8		(Optional) Signature

The *Signature* element in Table 17 is a signature over *SignedInfo* element described below in Table 18. The used signature algorithm is defined in *ContentAttestationRequest* message.

Table 18: SignedInfo Element for HMAC-SHA-256 Signature Type

# bytes	Type	Value	Description
16	Array of U8		Nonce as provided by the MirrorLink Client in Content Attestation Request (Table 19).
2	U16		SignedInfo flag that defines the attested content. The possible values are defined in Table 17.
32	Array of U8		(Optional) SHA-256 hash of context information pseudo encoding, as provided within the last framebuffer update (as defined in Table 17). Included if SignedInfo flag has bit 0 set.
32	Array of U8		(Optional) SHA-256 hash of framebuffer content, as provided with the last framebuffer update (as defined in Table 17). Included if SignedInfo flag has bit 1 set.
4	U32		(Optional) Number of framebuffer pixels sent since previous *ContentAttestationResponse* message (as defined in Table 17). 32-bit unsigned integer in network byte order. Included if SignedInfo flag has bit 2 set.

A MirrorLink Server, supporting Content Attestation, shall support the attestation of the Context Information Pseudo Encoding. It should support attestation of the number of pixels and the framebuffer content as well.

The *ContentAttestationRequest* message is given in Table 19.

Table 19: *ContentAttestationRequest* Message

# bytes	Type	Value	Description
1	U8	128	Message-type
1	U8	14	Extension-type
2	U16	20 + N	Payload length
16	Array of U8		Random Nonce
2	U16	*Bit*	*Defines, what shall be attested and included into the hash ('1' = include, '0' = do not include)*
		[0]	Include context information pseudo encoding, as defined in Table 17.
		[1]	Include last framebuffer update, as defined in Table 17.
		[2]	Include number of pixels sent since previous *ContentAttestationResponse* message, as defined in Table 17.
1	U8	*Value*	*Used signature type*
		0	No signature
		1	The signature algorithm is HMAC-SHA-256 signature. The signed data is defined in Table 17.

# bytes	Type	Value	Description
1	U8	*Value*	*Used session key*
		0	No session key included
		1	Random 128-bit symmetric session key that is encrypted using the application specific public key that was bound to attestation of VNC Server. The encryption is done according to RSAPKCS#1 v1.5 format. The session key is used from the MirrorLink Server in all subsequent *ContentAttestationResponse* messages, until a new key is provided in next Content Attestation Request.
N	Array of U8		(Optional) Session key. The Client shall set session key in the beginning of each session so that the Server does not have to remember previous session keys and mapping of these keys to different Client devices.

7.8 Framebuffer Blocking Notification

The *FramebufferBlockingNotification* message allows the MirrorLink Client to notify the MirrorLink Server, that certain framebuffer content delivered to it (as part of a *FramebufferUpdate* message) has been blocked.

The original VNC protocol does not provide a mechanism to inform the MirrorLink Server about the blocking. If the VNC Server was aware, it could take further actions. The *FramebufferNotification* message will provide that feedback, as shown in Figure 12.

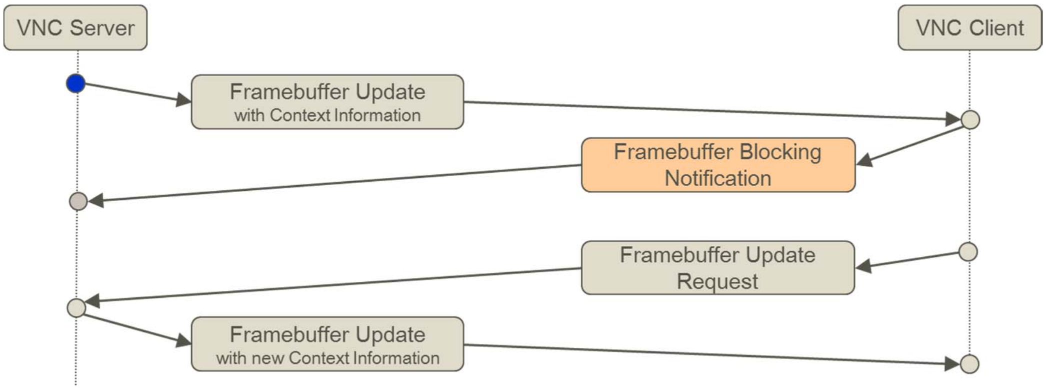

Figure 12: Example *FramebufferBlockingNotification* Message Flow

The *FramebufferBlockingNotification* message provides the MirrorLink Server with the rectangle area being blocked. This area and the application identifier shall correspond with the rectangle information from the *ContextInformationPseudoEncoding* message. The MirrorLink Client shall send a *FramebufferBlockingNotification* message for each rectangle, which is being blocked. I.e. if no rectangle is getting blocked, no *FramebufferBlockingNotification* message is sent.

The MirrorLink Client shall continue with sending *FramebufferBlockingNotification* messages, while it receives *FramebufferUpdate* messages, containing blocked rectangles.

The MirrorLink Server and Client have both shared responsibilities, when operating either in Immersive or Classic Mode, to prevent uncertified content showing up on the MirrorLink Client's screen. The target user experience to prevent and/or handle those situations is defined in [6]. Nevertheless, a MirrorLink Server and Client shall be able to handle even unexpected situations, e.g. non-MirrorLink application suddenly getting into the foreground. The MirrorLink Client shall not knowingly display not allowed content.

The following clause describes the different situations, framebuffer blocking can occur, and how MirrorLink Servers and Clients shall handle them.

MirrorLink Client is blocking the MirrorLink Foreground Application

The MirrorLink Client shall block non-Drive certified applications in Drive Mode.

NOTE: An exception for a limited set of non-certified applications from MirrorLink 1.0 Servers exist

The MirrorLink Client may block non-certified applications in Park Mode. The MirrorLink Client shall not block any Drive-certified application in Drive Mode, presented in landscape mode, and shall not block any Drive- or Base-certified application in Park Mode, presented in landscape mode.

In case of a blocking event, e.g. in case the MirrorLink Client switches from Park into Drive mode, or not allowed content is received, the following actions shall be taken:

Classic MirrorLink Mode

In *Classic MirrorLink* mode, the MirrorLink Client shall provide a user notification, containing the reason for the blocking action. The MirrorLink Client shall stop sending any further *FramebufferUpdateRequest* messages.

The MirrorLink Client shall execute one of the following two steps, dependent on which condition applies first:

- In case the user or the MirrorLink Client dismisses the notification, the MirrorLink Client shall move the MirrorLink Server's screen into the background, shall switch to the MirrorLink Client's MirrorLink application listing and shall send a *FramebufferBlockingNotification* message with the *reason-for-blocking* set to `"UI not visible on remote display"`. The MirrorLink Server shall inform the current foreground application via the MirrorLink API about the blocking.

NOTE: MirrorLink Client can automatically dismiss the notification without user intervention after a timeout, if sufficient time is available for the user to read the notification.

NOTE: The MirrorLink Client can monitor the UPnP *AppStatusUpdate* event, and then use the *GetApplicationStatus* action to determine, when a new application appears in the foreground. In case a blocking reason for that application does not exist, the MirrorLink Client can resume the VNC framebuffer transfer. The user notification can be removed, if the updated VNC context information contains the application, notified via the *AppStatusUpdate* event.

- In case the blocking reason disappears, e.g. the MirrorLink Client switches back into Park Mode or a drive-certified application gets to the foreground, while the notification is still shown, the MirrorLink Client shall immediately remove the user notification, shall move the MirrorLink Server's screen back into foreground, and shall resume sending *FramebufferUpdateRequest* messages.

NOTE: This allows blending in and out the blocking notification during Stop-and-Go traffic situations, while allowing the consumer to continue using the original application in Park Mode, without the need to relaunch it every time.

Despite the MirrorLink Server's screen move into the background, the MirrorLink Server and Client shall not intentionally terminate the VNC session.

A Legacy MirrorLink Client may send a *FramebufferBlockingNotification* message with the *reason-for-blocking* to `"Not sufficient application trust level"` (or other values). In response to this message, the MirrorLink Server shall send a *FramebufferUpdate* message, containing context information with at least one application category of `"Switch to MirrorLink Client native UI" (0xF000FFFF)`. The MirrorLink Client shall inform the user about the reason for the action. A Legacy Client or Server may terminate the VNC session.

Immersive MirrorLink Mode

In Immersive MirrorLink mode, the MirrorLink Server shall provide a user notification, containing the reason for the blocking action.

The MirrorLink Server shall execute one of the following two steps, dependent on which condition applies first:

- In case the user dismisses the notification, the MirrorLink Server shall move the MirrorLink application into the background and shall bring its Home Screen application into the foreground. The MirrorLink Server shall inform the current foreground application via the MirrorLink API about the blocking.
- In case the blocking reason disappears, e.g. the MirrorLink Client switches back into Park Mode, while the notification is still shown, the MirrorLink Server shall immediately remove the user notification and shall move the MirrorLink application back into foreground.

The MirrorLink Server and Client shall not intentionally terminate the VNC session.

In case the MirrorLink Client nevertheless receives not allowed content, it shall send a *FramebufferBlockingNotification* message with the *reason-for-blocking* set to `"Not sufficient application trust level"`. The MirrorLink Server shall respond to this message as described above.

In *Immersive MirrorLink* mode, the MirrorLink Server shall make use of the received *SetAllowedApplication* action of the UPnP Application Server service [4] to determine, which applications can be used within park and drive mode. The MirrorLink Server shall prevent the launch and appearance of any not allowed application in the respective Park and Drive Mode.

Any user notification shown during drive mode shall cover the full MirrorLink Screen to prevent any non-drive certified content being visible.

In case a MirrorLink Server fails to respond to the *FramebufferBlockingNotification* message appropriately, the MirrorLink Client shall switch to its native UI and inform the user about the blocking. It may take some time for the MirrorLink Server to remove a blocked application or overlay and bring another application to the foreground. Therefore, the MirrorLink shall allow for some grace period of at least 1s, prior taking other actions. While waiting, the MirrorLink Client shall continue sending *FramebufferUpdateRequest* messages.

MirrorLink Client is blocking a MirrorLink Overlay Frame

An overlay defines screen content, originating from a background application or from the platform, overlaying part (not all) of the current foreground application. I.e. the current foreground application is not changed. VNC context information of the overlay shall be provided in the context of the originating application.

The MirrorLink Client shall block overlays from non-drive certified applications in Drive Mode. The MirrorLink Client may block overlays from non-certified applications in Park Mode over the MirrorLink Server's current foreground application. The MirrorLink Client shall not block overlays from Drive-certified application in Drive Mode and shall not block overlays from Drive- or Base-certified application in Park Mode.

In case of a blocking event, the following actions shall be taken:

Classic MirrorLink Mode

In *Classic MirrorLink* mode, the MirrorLink Client shall send a *FramebufferBlockingNotification* message with the *reason-for-blocking* set to `"Not sufficient application trust level"`. In response to this message, the MirrorLink Server shall move the overlay to the background. Because the overlay comes from an application not visible to the user, the MirrorLink Client should not provide a user notification, it should discard the received framebuffer content and should continue showing the latest non-overlaid framebuffer.

In case the MirrorLink Server cannot remove the overlay, it shall provide context information, containing at least an application category of `"Switch to MirrorLink Client native UI" (0xF000FFFF)`. The MirrorLink Client shall inform the user about the reason for the action.

The MirrorLink Server and Client shall not intentionally terminate the VNC session.

Immersive MirrorLink Mode

In *Immersive MirrorLink* mode, The MirrorLink Server shall make use of the *SetAllowedApplication* action of the UPnP Application Server service to determine, which applications are allowed to be used within park and drive mode. The MirrorLink Server shall automatically remove overlays from not allowed applications.

In *Immersive MirrorLink* mode, the MirrorLink Client shall send a *FramebufferBlockingNotification* message with the *reason-for-blocking* set to `"Not sufficient application trust level"`. In response to this message, the MirrorLink Server shall move the overlay to the background.

Because the overlay comes from an application not visible to the user, the MirrorLink Server should not provide a user notification.

The MirrorLink Server and Client shall not intentionally terminate the VNC session.

In case the MirrorLink Server fails to respond to the *FramebufferBlockingNotification* message appropriately, the MirrorLink Client shall switch to its native UI and inform the user about the blocking. It may take some time for the MirrorLink Server to remove a blocked application or overlay and bring another application to the foreground. Therefore, the MirrorLink shall allow for some grace period of at least 1s, prior taking other actions. While waiting, the MirrorLink Client shall continue sending *FramebufferUpdateRequest* messages.

Implementation Notes:

The MirrorLink Server may be unable to detect Framebuffer overlays from background applications or platform functions, as detailed in the platform specific specifications (e.g. the Android-specific specifications are available in ETSI TS 103 544-22 [i.5]). In that case, the overlay will be reported as part of the current foreground application and stay in the foreground.

MirrorLink Server is switching the Framebuffer Orientation

The MirrorLink Client shall not block a framebuffer for the reason `"UI layout not supported"` in case the MirrorLink Server's framebuffer is in Landscape mode.

The MirrorLink Client shall block a framebuffer with the reason `"UI layout not supported"` in case the MirrorLink Server's framebuffer is in Portrait mode. The MirrorLink Server and Client and Server shall keep the VNC session alive.

Implementation Note:

MirrorLink 1.0, 1.1 and 1.2 can send a framebuffer in portrait orientation, but they will respond to the blocking notification by switching back to landscape.

MirrorLink Client is moving the MirrorLink Framebuffer to the Background

The MirrorLink Client may move the MirrorLink Server's framebuffer into the background on the MirrorLink Client display:

- In response to a consumer action, e.g. the consumer is launching a native application.
- In response to a framebuffer blocking situation, as described above.
- In response to the MirrorLink Server send context information with an application category of `"Switch to MirrorLink Client native UI"`.
- Other higher priority need for the MirrorLink Client to show a local user interface.

In such an event, the MirrorLink Client shall send a *FramebufferBlockingNotification* message with the *reason-for-blocking* set to `"UI not visible on remote display"`.

NOTE: Some older MirrorLink Clients may also additionally use the deprecated reason bit 8. The MirrorLink Server should ignore this reason bit.

When receiving the message, the MirrorLink Server shall not change the current foreground application. The current foreground application shall be notified via the MirrorLink API.

The VNC Client shall not send further *FramebufferUpdateRequest* messages. As soon as the MirrorLink Server's framebuffer becomes visible again on the MirrorLink Client display, the MirrorLink Client shall resume sending *FramebufferUpdateRequest* messages again. The MirrorLink Sever shall then notify the current foreground application via the MirrorLink API then again.

The MirrorLink Server and Client and Server shall keep the VNC session alive.

NOTE: Some older MirrorLink Clients or Server may terminate the VNC session, when the MirrorLink Server's Framebuffer is moved to the background. This does not have to be considered a misbehaving device.

Implementation Note - Exception:

Some existing MirrorLink 1.1 Servers may change the application state, when receiving a *FramebufferBlockingNotification* message for the reasons `"UI not visible on remote screen"`. A MirrorLink Client may therefore refrain from sending a *FramebufferBlockingNotification* message, when the MirrorLink Server's framebuffer is not visible. It shall refrain from requesting new *FramebufferUpdateRequest* message though. This exception is given only to MirrorLink Clients, connected to a MirrorLink 1.1 Server.

User Information about Framebuffer Blocking

The User shall be informed with a text message, pictogram or audio message explaining that the application is blocked, as indicated above. The user information should contain an indication about possible reasons for the blocking. For audio message, simple beeping sounds are not sufficient, a voice message indicating the blocking is needed.

In case notification is provided from the MirrorLink Server, the notification shall belong to an application drive-certified for the connected MirrorLink Client.

If blocking is occurring multiple times in a row (e.g. in slow moving traffic) the blocking message may be skipped after the first message/pictogram.

The *FramebufferBlockingNotification* message is given below.

Table 20: *FramebufferBlockingNotification* Message

# bytes	Type	Value	Description
1	U8	128	Message-type
1	U8	16	Extension-type
2	U16	14	Payload length
2	U16		X-position of rectangle (top left corner)
2	U16		Y-position of rectangle (top left corner)
2	U16		Width of rectangle
2	U16		Height of rectangle
4	U32		Unique application identifier (Shall be identical to the unique application ID provided within the *ContextInformationPseudoEncoding* message)
2	U16	*Bit*	Reason for blocking ('1' = reason, '0' no reason)
		[0]	*Not allowed content category* **Deprecated**. Shall be set to `"0"`.
		[1]	*Not allowed application category* **Deprecated**. Shall be set to `"0"`.
		[2]	*Not sufficient content trust level* **Deprecated**. Shall be set to `"0"`.
		[3]	*Not sufficient application trust level* Application is blocked for certification status reason (e.g. a non-drive application in drive mode).
		[4]	*Content rules not followed* **Deprecated**. Shall be set to `"0"`.

# bytes	Type	Value	Description
		[5]	*Not allowed application ID* **Deprecated**. Shall be set to "0".
		[8]	*UI not in focus on remote display* **Deprecated**. Shall be set to "0".
		[9]	*UI not visible on remote display* MirrorLink Client has moved the MirrorLink Server's framebuffer into the background.
		[10]	*UI layout not supported* Application is blocked because it switched into Portrait mode (after a Desktop Size Pseudo Encoding)

Implementation Note:

A MirrorLink 1.0, 1.1 or 1.2 Client may enable other reason bits for blocking as well. Those shall be ignored from MirrorLink 1.3 Servers. Any setting of bits 0, 1, 2, 4 or 5 shall be considered as bit 3 being set.

The *FramebufferBlockingNotification* message shall correspond to the last received *FramebufferUpdate* message with *ContextInformationPseudoEncoding*. The VNC Client shall send the *FramebufferBlockingNotification* message before the next *FramebufferUpdateRequest* message, in case the blocking event is in response to newly received framebuffer data. The VNC Client shall not wait for the next *FramebufferUpdate* message to arrive, in case the MirrorLink Client is switching from Park Mode into Drive Mode. The VNC Client shall send the VNC *FramebufferBlockingNotification* message immediately following the removal of the UI from its screen.

Handling of Overlays

The MirrorLink Server shall show overlays, only if one of the following conditions is true:

- Park mode is enabled.
- Application, creating the overlay, is drive-certified.
- Application, creating the overlay, is running in the foreground.
- Overlay is providing safety critical information, as outlined below.
- MirrorLink Server cannot detect overlays, as detailed in the platform specific specifications.
- MirrorLink Server cannot prevent applications to show overlays, as detailed in the platform specific specifications.

If available, the MirrorLink Server shall provide accurate context information, using the application identifier of the application creating the overlay; otherwise the MirrorLink Server shall use unknown values.

Handling of Applications Seeking Foreground Status

The MirrorLink Server shall allow an app to seek foreground status, only if one of the following conditions is true:

- Park mode is enabled.
- Application is drive-certified.
- Application is providing safety critical information, as outlined below.
- MirrorLink Server cannot prevent applications to go into the foreground, as detailed in the platform specific specifications.

Exceptional Safety Critical Content

A MirrorLink Server may display the following emergency alerts, as recommended by regional regulation, e.g. US FEMA [i.2], even if the content is not drive certified:

- Extreme weather, and other threatening emergencies in the local area:
  - Tsunami, earthquake warnings;
  - Tornado and flash flood warnings;
  - Hurricane, typhoon, dust storm and extreme wind warnings.
- Local law-enforcement alerts, e.g. AMBER Alerts [i.2].
- Presidential Alerts during a national emergency.

7.9 Audio Blocking Notification

The *AudioBlockingNotification* message allows the MirrorLink Client to notify the MirrorLink Server, that certain audio content delivered to it (as part of a RTP stream) has been blocked. In addition, the message can be used to resume the audio streaming of a previously blocked audio stream. These mechanisms do not apply for any Bluetooth connections, like BT HFP or BT A2DP.

The original VNC protocol does not provide a mechanism to inform the MirrorLink Server about the blocking. If the VNC Server was aware it could take further actions. The *AudioBlockingNotification* message will provide that feedback, as shown in Figure 13.

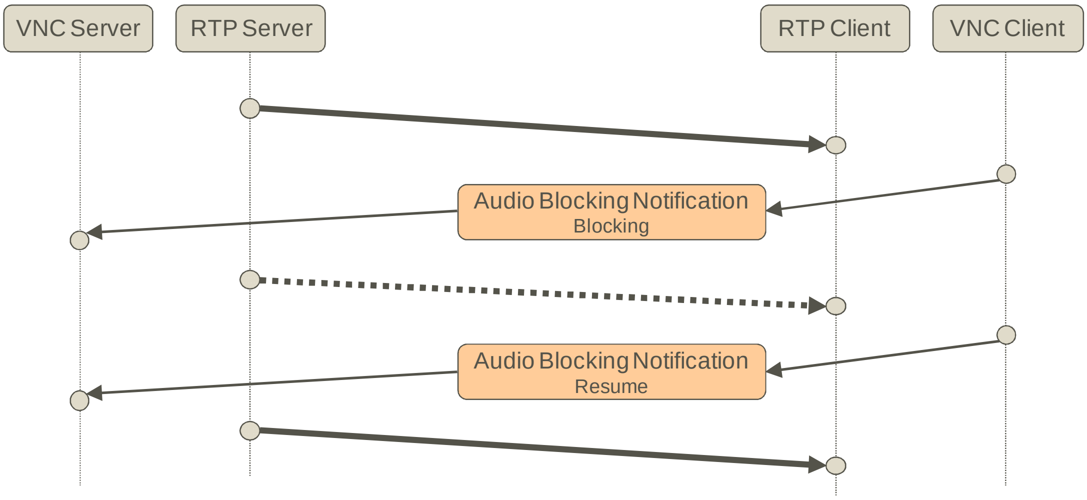

Figure 13: Example *AudioBlockingNotification* Message Flow

The *AudioBlockingNotification* message provides the MirrorLink Server with the application identifier whose stream will be blocked. The application identifier shall correspond to the advertised identifier in the UPnP application listing. The same application identifier shall be used within the RTP extension header, specified in [3]. The MirrorLink Client shall send an *AudioBlockingNotification* message for each audio stream being blocked.

MirrorLink Client is blocking the MirrorLink selected Audio Streams

The MirrorLink Client receives a single RTP stream, which may include audio from different audio sources. The MirrorLink Client may use the *AudioBlockingNotiifcation* message, to have the MirrorLink Server remove specific audio sources from the RTP stream (audio blocking). In order to block a particular audio stream, the MirrorLink Client shall send an *AudioBlockingNotifcation* messages for each RTP audio stream it wants to be removed.

The MirrorLink Client shall only block audio streams if higher-priority native MirrorLink Client audio is playing or the user intentionally muted the audio from an application.

On reception of an *AudioBlockingNotification* message, the MirrorLink Server shall respond with the following actions:

- The MirrorLink Server shall notify the application via the MirrorLink API about the blocking.
- The MirrorLink Server shall stop sending RTP packets containing foreground content from blocked applications.

NOTE: Foreground does not refer to the foreground application, but to the audio source, which is in the foreground within the audio stream (in case multiple audio sources are mixed together). The foreground audio source is always listed first in the RTP extension header.

NOTE: It is the responsibility of MirrorLink base or drive certified application to stop, pause or mute the current audio playback, when receiving an audio blocking callback via the Common API.

Implementation Notes:

The MirrorLink Server may be unable to remove audio from non-certified applications, as detailed in CCC's MirrorLink platform specific specifications (e.g. the Android-specific specifications are available in in ETSI TS 103 544-22 [i.5]). In that case, the audio will continue playing. A MirrorLink Server should attempt to terminate the respective application.

The MirrorLink Server and Client shall keep the RTP session and any VNC session alive.

NOTE: Some older MirrorLink Clients or Server may terminate the RTP or VNC session, when the MirrorLink Server's Framebuffer is moved to the background. This does not have to be considered a misbehaving device.

MirrorLink Client is globally muting all MirrorLink Audio Streams

The MirrorLink Client may block all audio stream from the MirrorLink Server (global mute), using the reason flag `"Global audio muted"`. The MirrorLink Client shall send *AudioBlockingNotification* messages for each Audio Source, reported via the RTP extension header. The MirrorLink Client may send *AudioBlockingNotification* messages to additional applications from the UPnP Application listing, which are currently not providing audio.

On reception of an *AudioBlockingNotification* message with global mute, the MirrorLink Server shall respond with the following actions:

- The MirrorLink Server shall notify applications via the MirrorLink API about the blocking.
- The MirrorLink Server shall stop sending RTP packets.

In case new audio sources come online on the MirrorLink Server, after the MirrorLink Client has globally muted the audio, the MirrorLink Server may resume sending RTP packets, as older MirrorLink Clients may NOT inform the MirrorLink Server about the global Audio unmute.

The MirrorLink Server and Client shall keep the RTP session and any VNC session alive.

MirrorLink Client is unblocking the MirrorLink selected or all Audio Streams

The MirrorLink Client shall send an *AudioBlockingNotification* message for each Audio Source it had previous blocked, with the reason flag being all zero, to indicate to the MirrorLink Server that an audio stream corresponding to the given application id is now unblocked. The MirrorLink Client may send *AudioBlockingNotification* messages with reason flag 0x00 to other applications from the UPnP Application listing, which had not been blocked before.

Note: Some older MirrorLink Client's need not send an *AudioBlockingNotification* message. In this case, the user will need to manually resume the audio playback.

On reception of an *AudioBlockingNotification* message, the MirrorLink Server shall respond with the following actions:

- The MirrorLink Server shall notify applications via the MirrorLink API about the unblocking.
- The MirrorLink Server shall resume sending RTP packets containing only content from unblocked applications, once audio becomes available again.

The *AudioBlockingNotification* message is given below in Table 21.

Table 21: *AudioBlockingNotification* Message

# bytes	Type	Value	Description
1	U8	128	Message-type
1	U8	18	Extension-type
2	U16	6	Payload length
4	U32		Unique application identifier If zero, this identifies RTP streams, belonging to applications, not being advertised individually.
2	U16	*Bit*	*Reason for blocking ('1' = reason, '0' no reason)*
		[0]	Not allowed application category **Deprecated**. Shall be set to "0".
		[1]	Not sufficient application trust level **Deprecated**. Shall be set to "0".
		[2]	Not allowed application ID **Deprecated**. Shall be set to "0".
		[3]	Global audio muted The MirrorLink Client shall set this reason bit for one of the following reasons: • MirrorLink Client is doing a Global audio mute.
		[4]	Audio stream, as given by application ID, muted The MirrorLink Client shall set this reason bit for one of the following reason: • User has muted the audio stream.

Implementation Note:

> A MirrorLink 1.0, 1.1 or 1.2 Client may enable other reason bits for blocking as well. Those shall be ignored from MirrorLink 1.3 Servers. Any setting of bits 0, 1, or 2 shall be considered as bit 4 being set.

The blocking of Audio from specific application can be modelled as if the MirrorLink Server is maintaining a blacklist of blocked applications, as identified by their *appIDs*. Applications are added to the blacklist, when an *AudioBlockingNotification* message is received, which indicates the blocking of an audio stream, otherwise the application is removed from the blacklist. On removal or addition, the affected application is always informed via a specific Common API callback. This is shown in Figure 14.

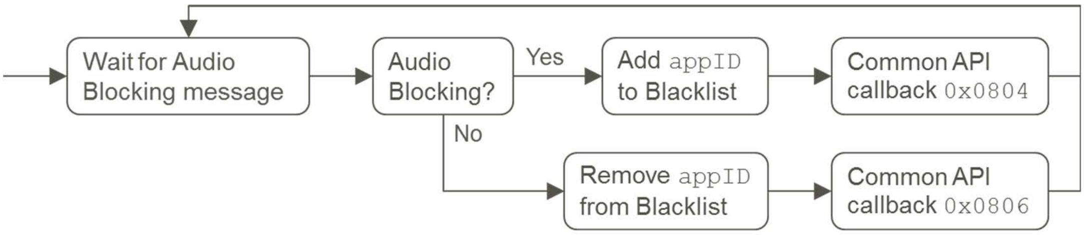
Wait for Audio Blocking message
Audio Blocking?
Yes
No
Add appID to Blacklist
Common API callback 0x0804
Remove appID from Blacklist
Common API callback 0x0806

Figure 14: Analysing the Audio Blocking Messages

The blacklist is cleared at the beginning of an RTP stream (Note: the end of an RTP stream is indicated through the Marker bit `M` being set to 1). Each incoming RTP packet is Analysed, whether its foreground audio (i.e. the first entry in the RTP extension header) is included in the blacklist. In that case, the RTP packet is discarded and not sent, otherwise it is sent to the MirrorLink Client's RTP Client. This behaviour is shown in Figure 15. Discarding of RTP packets will be only necessary if the MirrorLink Sever is not able to either remove the audio stream prior mixing or to terminate the application creating the blocked audio stream.

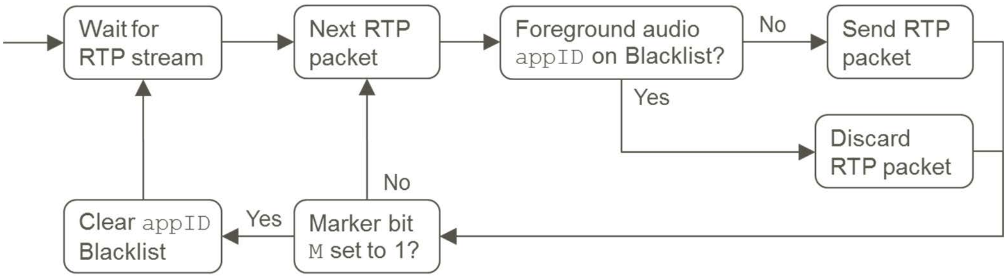

Figure 15: Analysing the Audio Packets

Above figures describe the main concept of how the MirrorLink Server shall handle the blocking of audio packets. Implementation details are implementation dependent. Some older MirrorLink Server may expose a different audio blocking behaviour.

7.10 Touch Event

The support for the *TouchEvent* message is optional. The MirrorLink Client should not send *TouchEvent* messages, if it has disabled Bit [1] of the *Pointer related* configuration vector in the *ClientEventConfiguration* message. The MirrorLink Server shall ignore *TouchEvent* messages, if the MirrorLink Server or Client has disabled Bit [1] of the *Pointer related* configuration vector in the Server or *ClientEventConfiguration* message.

The *TouchEvent* message allows the MirrorLink Client to notify the MirrorLink Server about a touch event. The original VNC protocol (version 3.8) provides support for Pointer Events only. The difference between pointer and touch events with regard to the present document is shown in Figure 16 as follows:

Pointer Event: Pointer events are used to describe touch screen action in which the user touches the screen with one (virtual) finger only at a single location.

Touch Event: Touch events are used to describe touch screen action in which the user touches the screen with one or more individual fingers at different locations. Touch events are used to describe more complex touch action, like pinch-open or pinch-close.

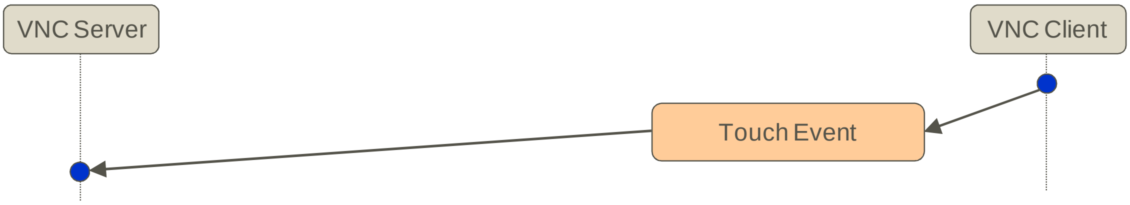

Figure 16: *TouchEvent* Message Flow

The *TouchEvent* message is given in Table 22.

Table 22: *TouchEvent* Message

# bytes	Type	Value	Description
1	U8	128	Message-type
1	U8	20	Extension-type
2	U16	1 + N*6	Payload length

# bytes	Type	Value	Description
1	U8	N	Number of individual events
N*6	Array of 6 bytes		Description of individual event

Each individual event is described in Table 23 the following way:

Table 23: Description of Individual Events

# bytes	Type	Value	Description
2	U16		X-position of the individual event
2	U16		Y-position of the individual event
1	U8		Event identifier
1	U8		Pressure value A zero value (0) indicates a touch release event, A non-zero value indicates a touch press event, with the given pressure level.

The VNC Client shall only use event identifier within the range `[0; Nmax-1]`, where `Nmax` is the maximum number of simultaneous supported touch events, as exchanged within the *ClientEventConfiguration* and *ServerEventConfiguration* pair message. Each event shall be completed, i.e. each press event shall be later followed by a release event.

The VNC Server shall ignore a touch release event, if no touch-press event has been received before for the same event identifier. The VNC Server shall complete an open touch-press event after 5 seconds if no touch-release event or no additional touch-press event has been received for the same event identifier.

The pressure value may be used from the VNC Client to indicate the pressure value received during the touch event. A value of 0x00 indicates as press release event, whereas the value given as the pressure mask, within the *ClientEventConfiguration* message, indicates the maximum pressure. The VNC Client should not provide pressure levels outside the Client's pressure mask. The VNC Server shall cap pressure levels at the Client's pressure mask. The VNC Server should adapt the received pressure level to its own pressure mask, e.g. using bit shift operations. I.e. a maximum pressure level at the Client should correspond to the maximum pressure level at the VNC Server.

Table 24 gives examples, how pressure values from the MirrorLink Client will be capped and adapted to match the supported Server pressure values, dependent on different configurations in the Server and *ClientEventConfiguration* messages.

NOTE: It is implementation specific, whether the MirrorLink Server will need to adapt the received pressure values.

Table 24: Adaptation of Touch Event Pressure Values

Server Event Pressure Mask	Client Event Pressure Mask	Pressure Value at Client	Value Capped at Client	Value Adapted at Server
0b00	Any value	No support for touch events from MirrorLink Server		
Any value	0b00	No support for touch events from MirrorLink Client		
0b01	0b01	0b00	0b00	0b00
		0b01	0b01	0b01
		>0b01	0b01	0b01

Server Event Pressure Mask	Client Event Pressure Mask	Pressure Value at Client	Value Capped at Client	Value Adapted at Server
0b11	0b111	0b000	0b000	0b00
		0b001	0b001	0b00
		0b010	0b010	0b01
		0b011	0b011	0b01
		0b100	0b100	0b10
		0b101	0b101	0b10
		0b110	0b110	0b11
		0b111	0b111	0b11
		>0b111	0b111	0b11
0b111	0b11	0b00	0b00	0b000
		0b01	0b01	0b010
		0b10	0b10	0b100
		0b11	0b11	0b110
		>0b11	0b11	0b110

Touch and pointer events shall not be mixed. A MirrorLink Client shall only use touch events, if the MirrorLink Sever and the MirrorLink Client supports it. Otherwise, pointer events shall be used.

8 Additional Encodings and Pseudo Encodings

8.1 General

Extensions to the VNC protocols include additional encodings and pseudo encodings. All new encodings are made in compliance with the RFB protocol. The additional encodings are summarized in Table 25.

Table 25: Additional VNC Encodings

Type	Value	Description	Functionality	Support
Pseudo Encoding	-523	MirrorLink Encoding	Advertise the support of MirrorLink extension messages. Not used within *FramebufferUpdate* messages.	Mandatory
Pseudo Encoding	-524	Context Information	Indicate context information within a *FramebufferUpdate* message	Mandatory
Pseudo Encoding	-223	Desktop Size	Change the VNC Server's framebuffer resolution.	Mandatory
Encoding	-525	Run-length-encoding	Scan line based run-length-encoding	Optional
Encoding	-526	Transform Encoding	Deprecated	Deprecated
Encoding	-527	HSML	HSML encoding	Optional
Encoding	0x48323634	VA H.264	H.264 encoding	Conditional (see note)
NOTE: Mandatory, if MirrorLink device supports a hardware H.264 encoder/decoder.				

The encodings are described in more detail in the following paragraphs.

8.2 MirrorLink Pseudo Encoding

The *MirrorLinkPseudoEncoding* is used within the *SetEncodings* message to inform the VNC Server about the Client support of the MirrorLink extension messages. The Client shall use *MirrorLinkPseudoEncoding* only within the *SetEncoding* message to indicate support for the MirrorLink extension messages. This Pseudo Encoding shall not be used within any *FramebufferUpdate* message.

The support for *MirrorLinkPseudoEncoding* is mandatory.

8.3 Context Information Pseudo Encoding

The *ContextInformationPseudoEncoding* is added to the framebuffer encoding types to provide the Client additional meta-information about the applications and content, copied via the Framebuffer Update messages. The context information can originate from different sources, giving different levels of trust in its correctness. If context information is available from different trust levels, the VNC Server shall provide the highest trust level to the Client.

The context information can be used within the VNC Client to make an informed decision, to what extent to bring the mobile device framebuffer content to the attention of the MirrorLink Client user. Dependent on legal considerations regarding driver distraction, part of the framebuffer content or all need not be shown. It is the responsibility of the VNC Client to make such decision. The VNC Server supports this decision process, providing accurate context information as shown in Figure 17.

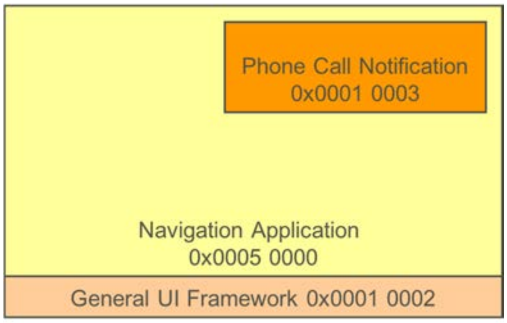

Figure 17: Context Information (Example)

Context information can be given either for the entire display at once, or for multiple, individual rectangular areas. Context information shall be valid, until the next context information pseudo encoding. The VNC Server shall provide context information for the entire display, i.e. the aggregation of the individual, related rectangular areas shall always cover the entire framebuffer, and never a partial framebuffer area alone.

> Drive certification of applications for MirrorLink currently requires single full-screen application within a MirrorLink session. Therefore, multiple rectangles shall be related, i.e. belonging to the same applications. Overlay notifications or application independent status bars may be included. To allow for future e.g. split-screen application, the MirrorLink Clients shall handle un-related context information rectangle.

If multiple overlapping rectangles are given, the sequence of the rectangles defines the stacking order (last rectangle on top). If the MirrorLink Client requests a non-incremental framebuffer update, the MirrorLink Server shall provide full context information at least for the requested rectangle, even if the context information has not changed from previous transfer.

For some part of the display, the application category need not be available, but the MirrorLink Server may be able to provide more information about the framebuffer content type or vice versa.

The *ContextInformationPseudoEncoding* follows the RFB protocol specification for encodings. Context Information Pseudo Encoding shall be provided together with the framebuffer data, within the same RFB *FramebufferUpdate* message. To allow efficient blocking at the VNC Client, the Context Information Pseudo Encoding shall be at the beginning of the RFB *FramebufferUpdate* message, i.e. before the actual framebuffer data.

The whole pseudo encoding rectangle is given in Table 26.

Table 26: Context Information Pseudo Encoding

# bytes	Type	Value	Description
2	U16		X-position of rectangle (top left corner)
2	U16		Y-position of rectangle (top left corner)
2	U16		Width of rectangle
2	U16		Height of rectangle
4	S32	-524	Encoding type
4	U32		Unique application id. For application being advertised via UPnP, the unique application id shall match the advertised *appID*. This field may be left empty (i.e. zero value).
2	U16		Trust Level for Application Category (see [4], Table A-1)
2	U16		Trust Level for Content Category (see [4], Table A-1) Shall be identical to the application category trust level.
4	U32		Application Category (see [4], Table A-2)
4	U32		Content Category (see [4], Table A-3)
4	U32	0x00	Content rules. **Deprecated**. Shall be set to `0x00000000`.

Driver distraction rules, which should be followed from applications running on the MirrorLink Server, are provided from the MirrorLink Client using the UPnP *TmClientProfile SetClientProfile* action, as specified in [5].

The VNC Server shall provide context information within the first RFB *FramebufferUpdate* message. Additionally, the VNC Server shall provide the context information whenever there is a change compared to the context from what was previously provided to the VNC Client.

If the information within the *ContextInformationPseudoEncoding* rectangle for an advertised application differs from the context information, provided within the UPnP application listing, the VNC Client shall use the information from the *ContextInformationPseudoEncoding* rectangle.

In *Classic MirrorLink* mode, if the VNC Client receives a *ContextInformationPseudoEncoding* message, with an application category set to `"Switch to MirrorLink Client native UI"` (0xF000FFFF) the MirrorLink Client shall switch to the native MirrorLink application listing. The MirrorLink Client shall not send a VNC *Framebuffer Blocking Notification* with a blocking reason other than `"UI not in foreground"`.

In *Immersive MirrorLink* mode, if the VNC Client receives a *ContextInformationPseudoEncoding* message, with an application category set to `"Switch to MirrorLink Client native UI"` (0xF000FFFF) the MirrorLink Client shall switch to a native user-interface. The MirrorLink Client shall not send a VNC *Framebuffer Blocking Notification* with a blocking reason other than `"UI not in foreground"`.

8.4 Desktop Size Pseudo Encoding

The *DesktopSizePseudoEncoding* rectangle within a *FramebufferUpdate* message indicates to the VNC Client, that the VNC Server has changed its framebuffer resolution. The format of *DesktopSizePseudoEncoding* is specified in the RFB specification [1]. A *FramebufferUpdate* message, containing a *DesktopSizePseudoEncoding* rectangle shall not include Content Information Rectangle(s) and Framebuffer Data rectangle(s), even if the VNC Client originally requested a non-incremental Framebuffer Update.

Implementation Note for MirrorLink 1.0 to 1.2 Clients:

> MirrorLink 1.0 to 1.2 Clients may include framebuffer and context information data in framebuffer updates, containing a Desktop Size Pseudo Encoding rectangle. This data shall be ignored.

The MirrorLink Client shall support Desktop Size Pseudo Encoding.

The MirrorLink Client may request a new framebuffer update, before the last *FramebufferUpdate* message has been fully decoded. Therefore, the VNC Client may miss a *DesktopSizePseudoEncoding* rectangle, and may therefore request a Framebuffer Update for the "old" framebuffer dimension.

Therefore, the MirrorLink Server shall send a *FramebufferUpdate* message with a *DesktopSizePseudoEncoding* message, if the MirrorLink Client has requested a Framebuffer Update with a framebuffer area exceeding the MirrorLink Server's framebuffer resolutions.

8.5 Scan Line based Run-Length Encoding

Scan line based run-length encoding scans the identified rectangular framebuffer update area for a run of identical pixel values. Scanning shall be done for each row individually from the top row to the bottom row and from left-to-right within each row. Each run describes a number of pixels having the same colour value. A run shall not span across multiple lines. If the length of a run is bigger than maximum allowed run-length, the run shall be split into smaller runs. The minimum run length is 1. The whole Run-Length Encoding is given in Table 27.

The basic structure is defined in the RFB *FramebufferUpdate* message.

Table 27: Run-Length Encoding within a Framebuffer Update Message

# bytes	Type	Value	Description
2	U16		X-position of rectangle (top left corner)
2	U16		Y-position of rectangle (top left corner)
2	U16		Width of rectangle
2	U16		Height of rectangle
4	S32	-525	Run-Length Encoding type
N	Array of U8		Run-Length encoded data, encoded line-by-line, following the encoding format given in Table 28.

The run-length encoding shall be done line-by-line. The encoding for each individual line is given in the Table 28. Runs shall not span across multiple lines.

Table 28: Run-Length Encoding for individual Line

# bytes	Type	Value	Description
2	U16	M	Number of Runs within Line (In network endian)
M * K	Array of Run-length Encodings		M blocks of Run-Length encoded data, each having K bytes. K is dependent on the Colour depth

The number of bytes per run (K) is dependent on the colour depth of the selected pixel format. The run-length encoding only considers the colour value of the pixel. If the bits-per-pixel value is higher than the colour depth, those additional bits are ignored. The encoding of an individual run is shown in Table 29.

Table 29: Run-Length Encoding for Individual Run

# bytes	Type	Value	Description
K	Run-length Encoding	*Bit*	*Run-length Encoding*
		[C+R-1:C]	Run length minus 1
		[C-1:0]	Colour Value

NOTE: Endianess of the K-byte block follows the pixel's colour value endianess, as set from the VNC Server (RFB *ServerInit* message) or requested from the VNC Client (*SetPixelFormat* message).

NOTE: Colour-value contains bits of the pixel value; E.g. in case of ARGB888, these are 24 bits containing the R, G and B values; transparency bits are not included.

The number of bits C, representing the colour value of the run, shall be identical with the colour depth of the selected pixel format. The number of bits R representing the run-length minus 1 shall be according the following formula:

$$R = \begin{cases} 8 - C_m, & C_m \le 4 \\ 16 - C_m, & C_m > 4 \end{cases}$$

$$C_m = C \bmod 8$$

$$K = (R + C)/8$$

The maximum run-length is therefore at least 16 (4 bits). Table 30 gives run-length encoding values for example colour formats.

Table 30: Example Run-Length Encoding Values

Colour Format	Bits per Pixel	Colour Depth	# bytes K	# bits for colour value C	# bits for run length R
ARGB 888	32	24	4 (U32)	24	8
RGB 565	16	16	3 (U24)	16	8

8.6 VA H.264 Encoding

8.6.1 Overview

VA H.264 is an H.264 RFB encoding registered with the `0x48323634` encoding number at the IANA [i.3], which features H.264 compression of framebuffers that are sent from an RFB server to an RFB client. Using H.264 offers significant advantages like being able to leverage hardware based encoding and decoding, and being suitable to transmit high resolution, high frame rates video stream using a relatively low bandwidth.

H.264 based encoding differs from other lossless RFB encoding due to the various technique employed to achieve a high quality low bandwidth video stream between server and client which affect both the spatial and temporal properties of the stream.

Using H.264, framebuffers can be encoded in isolation using spatial compression techniques (thus reducing the overall size of the total data that has to be transmitted) or in reference to a previous framebuffer using temporal compression techniques (thus further reducing the size of the data) which are then embedded in framebuffer updates. Whereas H.264 mainstream usage is the streaming of media content, it is possible to use it in a non-streaming context (such as VNC) providing the guidelines presented hereafter are followed.

The basic structure is defined in the RFB *FramebufferUpdate* message.

Table 31: VA H.264 Encoding within a Framebuffer Update Message

# bytes	Type	Value	Description
2	U16		X-position of rectangle (top left corner)
2	U16		Y-position of rectangle (top left corner)
2	U16		Width of rectangle
2	U16		Height of rectangle
4	S32	0x48323634	VA H.264 Encoding type
4	U32		Size in bytes of the embedded H.264 Access Unit (*N*). May be zero for a No-Operation frame.
4	U32	*Value*	*Type of embedded H.264 Access Unit*
		0	Predicted Frame (P-Frame)
		1	Bi-directional-Predicted Frame (B-Frame)
		2	Intra Frame prepended with SPS/PPS Non VCL blocks (I-Frame)
4	U32		Width in pixels of the encoded H.264 frame, may exceed the FBU rectangle size, shall not exceed the Server's negotiated resolution

# bytes	Type	Value	Description
4	U32		Height in pixels of the encoded H.264 frame, may exceed the FBU rectangle size, shall not exceed the Server's negotiated resolution
N	U8		H.264 Access Unit, using Network Abstraction Layer (NAL) Units based transport, in Byte-Stream Format as defined in Annex B of [8].

8.6.2 Theory of operation

A VNC *FramebufferUpdate* message may contain multiple rectangles. If multiple rectangles are generated for a single server framebuffer update using the VA H.264 encoding, the H.264 Access Unit shall be sent only in the first rectangle of the *FramebufferUpdate* message. Subsequent rectangles shall contain only the coordinates and size of the additional updated regions, with the Size in bytes of the H.264 access unit set to 0. This allows the VNC Server to provide the VNC Client with additional fine-grain information of updated framebuffer areas.

A valid framebuffer update VA H.264 rectangle shall comprise of a full access unit, which shall be formatted using NAL Units in Byte-Stream format as defined in Annex B of [8], where the embedded Access Unit is either an Intra-Coded frame, a Bidirectional-predicted frame or a Predicted-frame. To guarantee the best interoperability with different range/grade of hardware, only I- and P- frame shall be sent when using VA H.264 as part of a MirrorLink connection. B-frames shall not be used. This is consistent with the use of the Baseline Profile (Designed primarily for lower-cost applications with limited computing resources, this profile is used widely in videoconferencing and mobile application).

A MirrorLink Client supporting the VA H.264 Encoding, shall support Baseline Profile at level 3.1. MirrorLink Clients, supporting VA H.264 Encoding and announcing a MirrorLink Client framebuffer resolution exceeding 1 280 x 720 shall support Baseline Profile at level 4.1. Profiles and levels are defined in Annex A of [8].

A MirrorLink Server may encode a slightly larger framebuffer, than the original framebuffer adding a couple of rows or column for more efficient encoding (e.g. to ensure that encoding width and height are a multiple of 16). In such case, the MirrorLink Server shall add H.264 cropping flags to the H.264 parameter set. MirrorLink Clients shall remove crop framebuffer data in accordance with the H.264 cropping flags included by the server. The MirrorLink Server shall advertise the useful (i.e. the non-cropped area) in its VNC *ServerInit* message or any subsequent *DesktopSizePseudoEncoding* rectangle within a *FramebufferUpdate* message.

The first H.264 Access Unit being transmitted in a run shall be an I-Frame prepended with the parameter set using Non VCL NAL Units (SPS/PPS).

Since, RFB is using TCP and not UDP and packet loss will not occur, all subsequent frames can be P-frames until the current H.264 run is broken by the server by either switching to another encoding or by changing the geometry of the framebuffer using the *DesktopSizePseudoEncoding* rectangle within a *FramebufferUpdate* message. When the *FramebufferUpdate* message includes a *DesktopSizePseudoEncoding* rectangle, the VA H.264 rectangle may contain a no-op (i.e. zero size) framebuffer, allowing both Server and Client to reconfigure the H.264 codecs.

Since H.264 has both Spatial and Temporal properties, special care has to be taken when using non-incremental and incremental framebuffer update request. The MirrorLink Server shall respond to a non-incremental *Framebuffer Update Request* message with an I-frame.

The Server shall provide an H.264 access unit immediately upon receiving a non-incremental Framebuffer Update Request.

A MirrorLink Client can implement a decoding pipeline, which introduces a decoding latency of one or more received framebuffer data updates. To ensure, that a quasi-static application UI, like a clock, is visible without additional latency, the MirrorLink Client will need to receive additional framebuffer data updates, to advance its decoding pipeline. Therefore, The MirrorLink Server shall continue responding to at least 5 additional (incremental) *Framebuffer Update Request* messages with further *FramebufferUpdate* messages, even if the Server's framebuffer content itself has not changed anymore The MirrorLink Client's decoding pipeline shall be equal or less than 5 stages. This mechanism is meant to provide MirrorLink Clients with timely updates of slowing changing user interfaces; it is not a mechanism to oversample an already fluid user interface. Therefore, the MirrorLink Server and/or Client should restrict the update rate to 30 fps.

The Server shall not drop any frames emitted by the encoder from the stream presented to the client, and the Client shall not drop any frames sent to the decoder from the stream received from the server. Frame dropping if necessary shall be implemented before the encoder and after the decoder.

The general flow of H.264 encoded Framebuffer Updates, with incremental and non-incremental framebuffer updates is shown Figure 18.

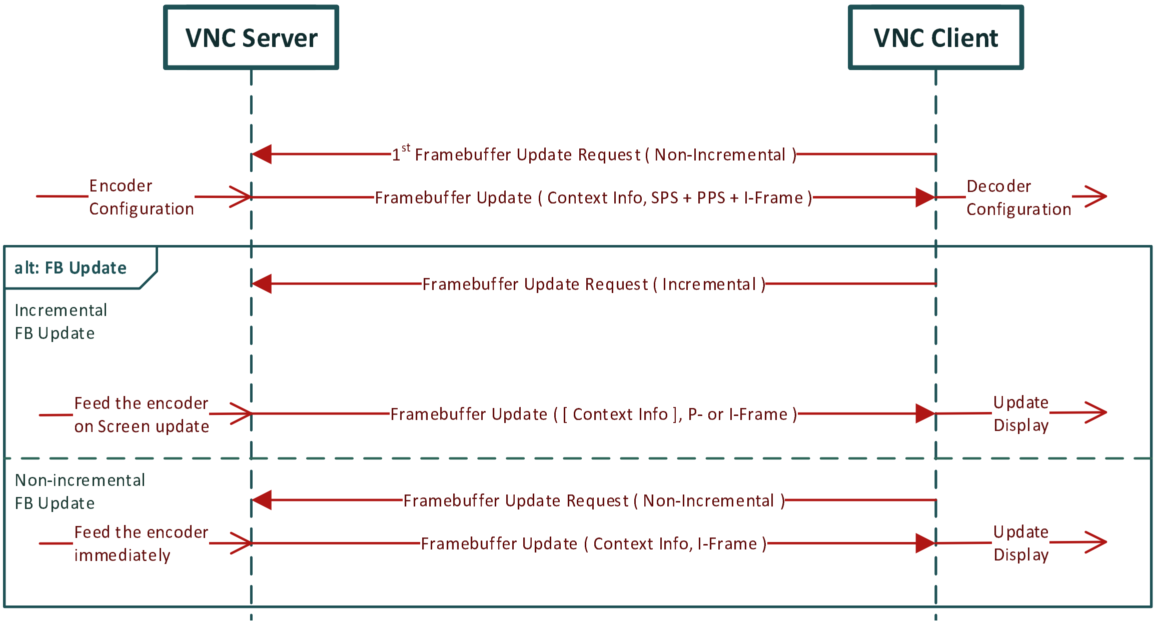
VNC Server
VNC Client
1st Framebuffer Update Request (Non-Incremental)
Encoder Configuration
Framebuffer Update (Context Info, SPS + PPS + I-Frame)
Decoder Configuration
alt: FB Update
Incremental FB Update
Framebuffer Update Request (Incremental)
Feed the encoder on Screen update
Framebuffer Update ([Context Info], P- or I-Frame)
Update Display
Non-incremental FB Update
Framebuffer Update Request (Non-Incremental)
Feed the encoder immediately
Framebuffer Update (Context Info, I-Frame)
Update Display

Figure 18: H.264 encoded (non-)Incremental Framebuffer Update - Message Flow

In case the VNC Client loses synchronization with the incoming framebuffer data stream, it shall send a *ClientDisplayConfiguration* message, containing the identical framebuffer resolution as sent within the previous *ClientDisplayConfiguration* message, followed by a non-incremental *FramebufferUpdateRequest* message. The VNC Server shall reset the H.264 encoder, i.e. flush the encoder's pipeline, and start fresh. An example message flow in shown in Figure 19.

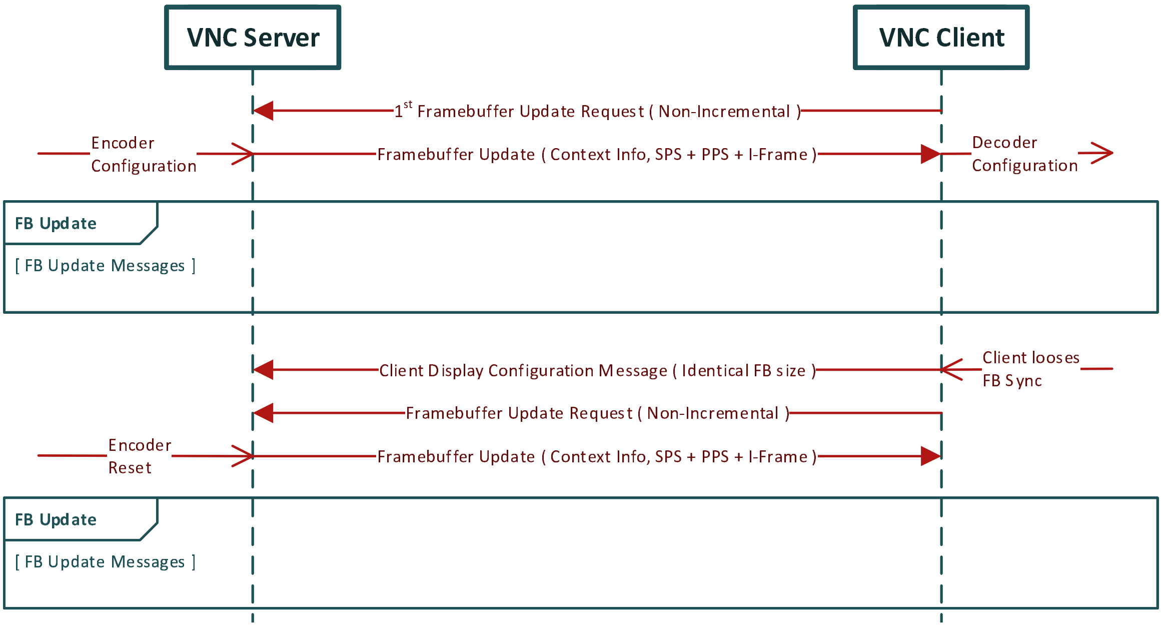
VNC Server
VNC Client
1st Framebuffer Update Request (Non-Incremental)
Encoder Configuration
Framebuffer Update (Context Info, SPS + PPS + I-Frame)
Decoder Configuration
FB Update
[FB Update Messages]
Client Display Configuration Message (Identical FB size)
Client looses FB Sync
Framebuffer Update Request (Non-Incremental)
Encoder Reset
Framebuffer Update (Context Info, SPS + PPS + I-Frame)
FB Update
[FB Update Messages]

Figure 19: VNC Client loses Synchronization of received Framebuffer - Message Flow

In case the VNC Client changes its framebuffer resolution (resizing), it shall send a *ClientDisplayConfiguration* message, containing the new framebuffer resolution, followed by an incremental *FramebufferUpdateRequest* message, containing the old resolution. The VNC Server shall send a *FramebufferUpdate* message, containing only a *DesktopSizePseudoEncoding* rectangle with the new framebuffer resolution. The VNC Client shall then send a non-incremental *FramebufferUpdateRequest* message. The VNC Server shall reconfigure the H.264 encoder and start fresh. An example message flow in shown in Figure 20.

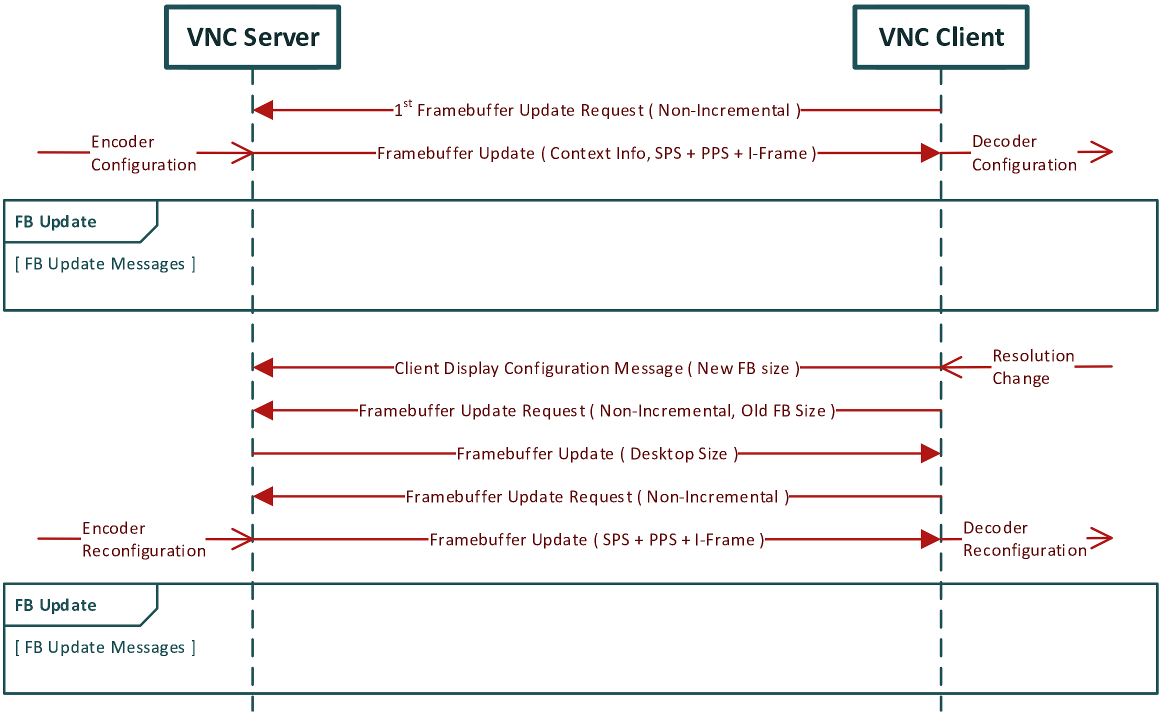
VNC Server
VNC Client
1st Framebuffer Update Request (Non-Incremental)
Encoder Configuration
Framebuffer Update (Context Info, SPS + PPS + I-Frame)
Decoder Configuration
FB Update
[FB Update Messages]
Client Display Configuration Message (New FB size)
Resolution Change
Framebuffer Update Request (Non-Incremental, Old FB Size)
Framebuffer Update (Desktop Size)
Framebuffer Update Request (Non-Incremental)
Encoder Reconfiguration
Framebuffer Update (SPS + PPS + I-Frame)
Decoder Reconfiguration
FB Update
[FB Update Messages]

Figure 20: VNC Client changes the Framebuffer Resolution - Message Flow

In case the VNC Server changes its framebuffer resolution (resizing), it shall send a *FramebufferUpdate* message, containing only a *DesktopSizePseudoEncoding* rectangle with the new framebuffer resolution. The VNC Client shall then send a non-incremental *FramebufferUpdateRequest* message. The VNC Server shall reconfigure the H.264 encoder and start fresh. An example message flow in shown in Figure 21.

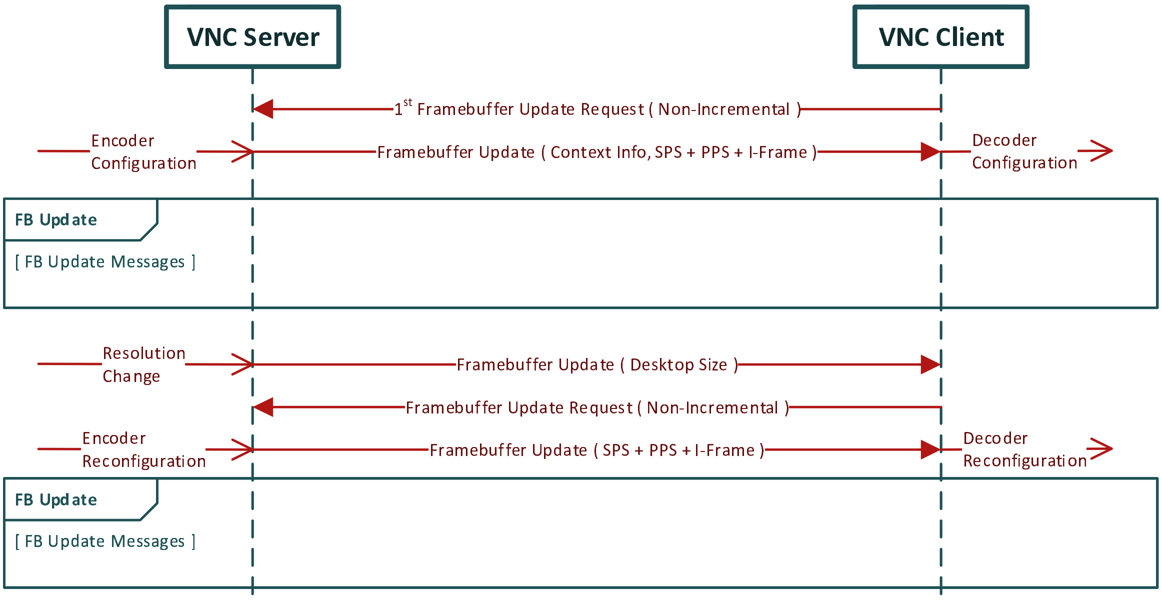
VNC Server
VNC Client
1st Framebuffer Update Request (Non-Incremental)
Encoder Configuration
Framebuffer Update (Context Info, SPS + PPS + I-Frame)
Decoder Configuration
FB Update
[FB Update Messages]
Resolution Change
Framebuffer Update (Desktop Size)
Framebuffer Update Request (Non-Incremental)
Encoder Reconfiguration
Framebuffer Update (SPS + PPS + I-Frame)
Decoder Reconfiguration
FB Update
[FB Update Messages]

Figure 21: VNC Server changes the Framebuffer Resolution - Message Flow

In case the VNC Client blocks in the incoming framebuffer, it shall send a *FramebufferBlockingNotification* message. The VNC Server shall stop feeding the encoder. As soon as the VNC Client resumes the framebuffer transfer, sending a *FramebufferBlockingNotification* message, the VNC Server shall resume feeding the encoder and shall send a regular FramebufferUpdate message, containing either a P or I-Frame. The MirrorLink Client shall keep the latest state of the framebuffer data, during blocking. An example message flow in shown in Figure 22.

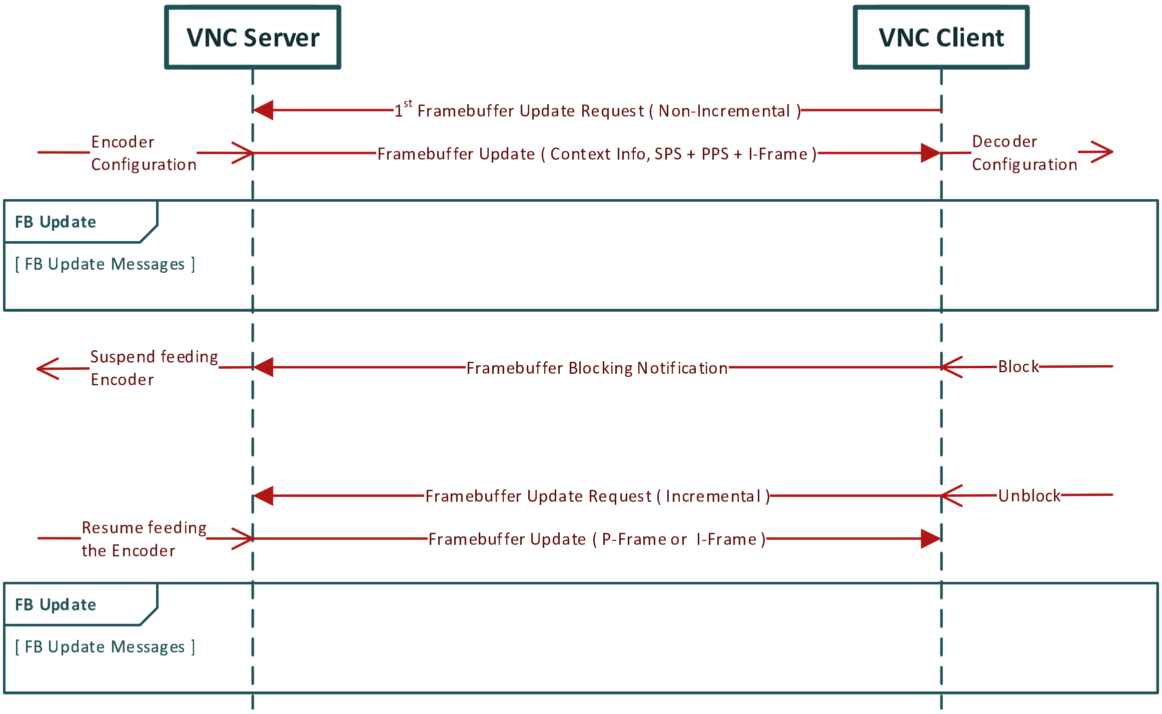
VNC Server
VNC Client
1st Framebuffer Update Request (Non-Incremental)
Encoder Configuration
Framebuffer Update (Context Info, SPS + PPS + I-Frame)
Decoder Configuration
FB Update
[FB Update Messages]
Suspend feeding Encoder
Framebuffer Blocking Notification
Block
Framebuffer Update Request (Incremental)
Unblock
Resume feeding the Encoder
Framebuffer Update (P-Frame or I-Frame)
FB Update
[FB Update Messages]

Figure 22: VNC Client blocks the Framebuffer while keeping State - Message Flow

In case the VNC Client loses state, i.e. it cannot keep the latest state of the framebuffer data, it shall reset encoder by sending a *ClientDisplayConfiguration* message, containing the identical framebuffer resolution as sent within the previous *ClientDisplayConfiguration* message, followed by a non-incremental *FramebufferUpdateRequest* message. The VNC Server shall reset the H.264 encoder, i.e. flush the encoder's pipeline, and start fresh. An example message flow in shown in Figure 23.

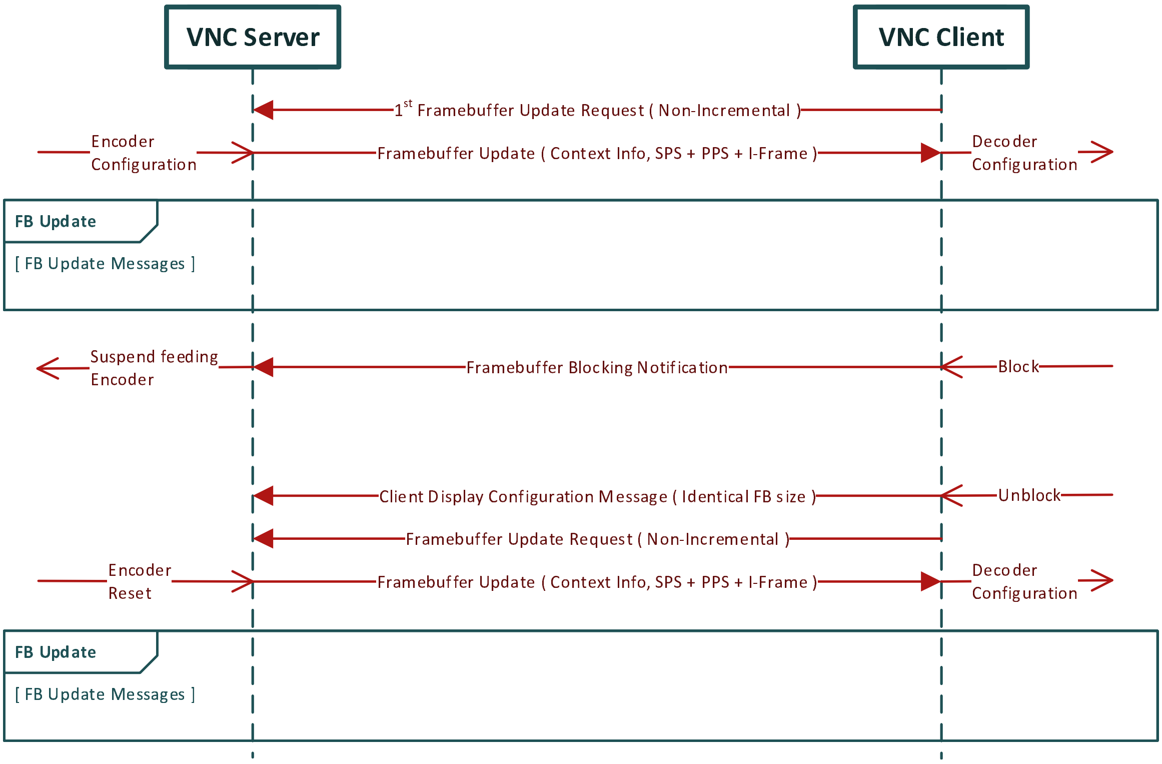
VNC Server
VNC Client
1st Framebuffer Update Request (Non-Incremental)
Encoder Configuration
Framebuffer Update (Context Info, SPS + PPS + I-Frame)
Decoder Configuration
FB Update
[FB Update Messages]
Suspend feeding Encoder
Framebuffer Blocking Notification
Block
Client Display Configuration Message (Identical FB size)
Unblock
Framebuffer Update Request (Non-Incremental)
Encoder Reset
Framebuffer Update (Context Info, SPS + PPS + I-Frame)
Decoder Configuration
FB Update
[FB Update Messages]

Figure 23: VNC Client blocks the Framebuffer while losing State - Message Flow

In case the VNC Server using H.264 encoding and wants to switch to a non-H.264 encoding, it shall send a *FramebufferUpdate* message, containing the framebuffer data, encoded with the new non-H.264 encoding, like RAW or scan-line based run-length encoding (RLE). The VNC Client shall tear down its H.264 decoder and shall start using the received encoding scheme.

In case the Server VNC Server is using non-H.264 encoding and wants to switch to H.264 encoding, it shall send a *FramebufferUpdate* message, containing the framebuffer data, encoded with H.264 encoding. The first H.264 Access Unit being transmitted shall be an I-Frame prepended with the parameter set using Non VCL NAL Units (SPS/PPS). The VNC Client shall configure its decoder appropriately.

An example message flow in shown in Figure 24.

The VNC Server shall not mix H.264 encoded rectangles with non-H.264 encoded rectangles within the same *FramebufferUpdate* message. The VNC Server should not frequently switch back and forth between H.264 and non-H.264 encoding.

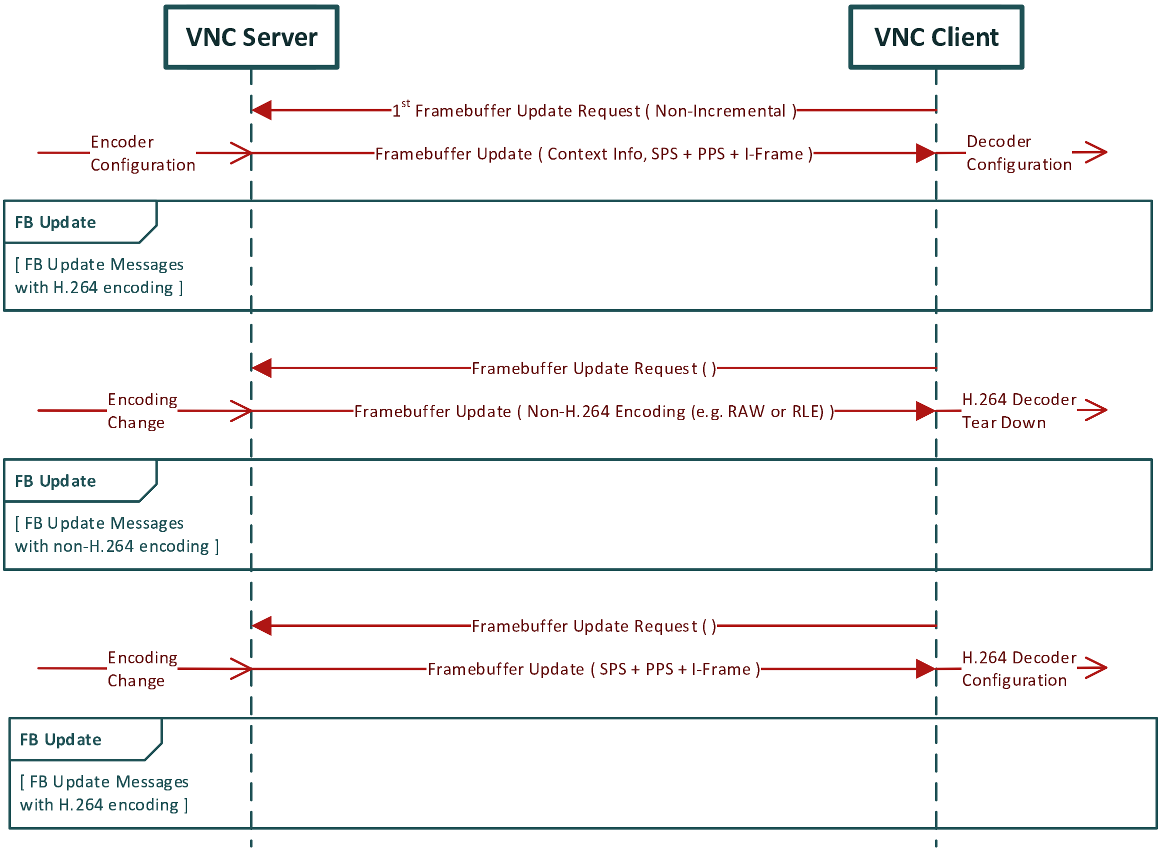
VNC Server
VNC Client
1st Framebuffer Update Request (Non-Incremental)
Encoder Configuration
Framebuffer Update (Context Info, SPS + PPS + I-Frame)
Decoder Configuration
FB Update
[FB Update Messages with H.264 encoding]
Framebuffer Update Request ( )
Encoding Change
Framebuffer Update (Non-H.264 Encoding (e.g. RAW or RLE))
H.264 Decoder Tear Down
FB Update
[FB Update Messages with non-H.264 encoding]
Framebuffer Update Request ( )
Encoding Change
Framebuffer Update (SPS + PPS + I-Frame)
H.264 Decoder Configuration
FB Update
[FB Update Messages with H.264 encoding]

Figure 24: VNC Server changes Framebuffer Encoding - Message Flow

Annex A (normative): Knob Configuration

The shift and rotate operations of a 2D knob are given according to the coordinate system, given in Figure A.1. The grey 2D area references the surface the knob is mounted to.

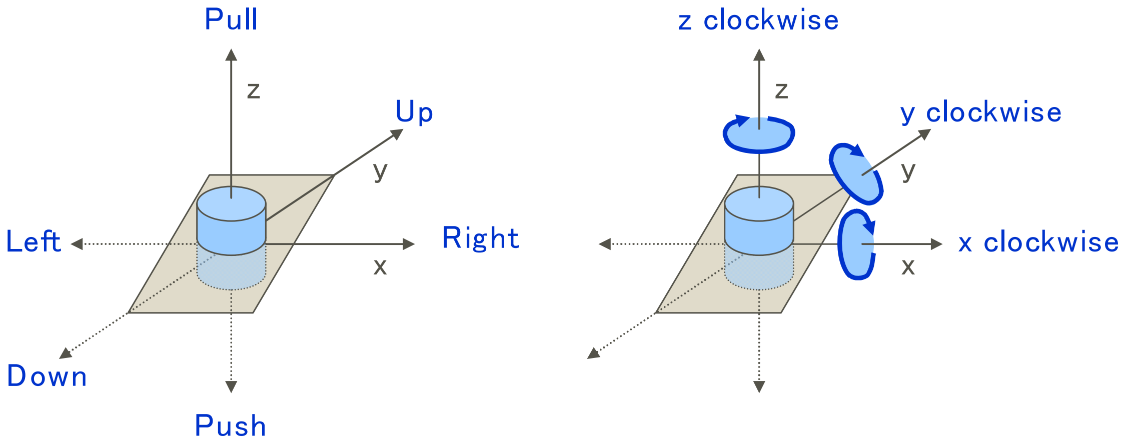

Figure A.1: Coordinate System for Knob Events

The knob shift & rotate configuration is shown in Table A.1, with allowed values for n in [0:3].

Table A.1: Knob Shift and Rotate Configuration Settings

# bytes	Type	Value	Description
4	U32	*Bit*	*Knob shift & rotate configuration (1 = support, 0 = no support)*
		[n*8 + 0]	Knob n: Shift along x axis
		[n*8 + 1]	Knob n: Shift along y axis
		[n*8 + 2]	Knob n: Shift along xy diagonals
		[n*8 + 3]	Knob n: Push along z axis
		[n*8 + 4]	Knob n: Pull along z axis
		[n*8 + 5]	Knob n: Rotation around x axis
		[n*8 + 6]	Knob n: Rotation around y axis
		[n*8 + 7]	Knob n: Rotation around z axis

The 32-bit configuration vector can hold up to 4 knob configurations. The MirrorLink Client and Server shall sequentially fill the configuration vector from the beginning (i.e. n = 0).

The MirrorLink Server may support long key-press events, i.e. multiple key-press events, before the final key-release event. The long key-press includes knob rotation events. If a knob provides haptic feedback, while rotating, it may give better user experience not to use long press events, but rather individual per click events (i.e. key-press event, followed by key-release event).

Annex B (normative): Key Event Mapping

The Key event mapping for different 2D knobs is shown in Table B.1. The key event mapping for a particular head-unit knob n shall be according the following format:

```
0x3000 00nm
```

The value n defines the head-unit knob and m defines the event as defined in the template above. Allowed values for n are [0:3] and for **m** are [0:F].

Table B.1: Head-Unit Knob Key Event Mapping

Category	Mnemonic	KeySymValue	Description
Knob Keys	Knob_2D_n_shift_right	0x3000 00n0	Right shift
	Knob_2D_n_shift_left	0x3000 00n1	Left shift
	Knob_2D_n_shift_up	0x3000 00n2	Up shift
	Knob_2D_n_shift_up_right	0x3000 00n3	Up & right shift
	Knob_2D_n_shift_up_left	0x3000 00n4	Up & left shift
	Knob_2D_n_shift_down	0x3000 00n5	Down shift
	Knob_2D_n_shift_down_right	0x3000 00n6	Down & right shift
	Knob_2D_n_shift_down_left	0x3000 00n7	Down & left shift
	Knob_2D_n_shift_push	0x3000 00n8	Push
	Knob_2D_n_shift_pull	0x3000 00n9	Pull
	Knob_2D_n_rotate_x	0x3000 00nA	x clockwise rotation
	Knob_2D_n_rotate_X	0x3000 00nB	x anti-clockwise rotation
	Knob_2D_n_rotate_y	0x3000 00nC	y clockwise rotation
	Knob_2D_n_rotate_Y	0x3000 00nD	y anti-clockwise rotation
	Knob_2D_n_rotate_z	0x3000 00nE	z clockwise rotation
	Knob_2D_n_rotate_Z	0x3000 00nF	z anti-clockwise rotation
	Reserved	0x3000 0040	-
		…	
		0x3000 00FF	

The Key event values for the ITU, Device, Function and Multimedia key events are given in Table B.2.

Table B.2: Key Event Mapping

Category	Mnemonic	KeySymValue	Description
ITU Keys	ITU_Key_0	0x3000 0100	0, ' '
	ITU_Key_1	0x3000 0101	1, '.', ','
	ITU_Key_2	0x3000 0102	2, a, b, c
	ITU_Key_3	0x3000 0103	3, d, e, f
	ITU_Key_4	0x3000 0104	4, g, h, i
	ITU_Key_5	0x3000 0105	5, j, k, l
	ITU_Key_6	0x3000 0106	6, m, n, 0
	ITU_Key_7	0x3000 0107	7, p, q, r, s
	ITU_Key_8	0x3000 0108	8, t, u, v
	ITU_Key_9	0x3000 0109	9, w, x, y, z
	ITU_Key_Asterix	0x3000 010A	*, +
	ITU_Key_Pound	0x3000 010B	#, shift
	Reserved	0x3000 010C	-
		…	
		0x3000 01FF	
Device Keys	Device_Phone_call	0x3000 0200	Take a phone call
	Device_Phone_end	0x3000 0201	End phone call
	Device_Soft_left	0x3000 0202	Left soft key
	Device_Soft_middle	0x3000 0203	Middle soft key
	Device_Soft_right	0x3000 0204	Right soft key
	Device_Application	0x3000 0205	Shortcut to the Application listing
	Device_Ok	0x3000 0206	Ok
	Device_Delete	0x3000 0207	Delete (Backspace)
	Device_Zoom_in	0x3000 0208	Zoom in
	Device_Zoom_out	0x3000 0209	Zoom out
	Device_Clear	0x3000 020A	Clear
	Device_Forward	0x3000 020B	Go one step forward
	Device_Backward	0x3000 020C	Go one step backward
	Device_Home	0x3000 020D	Shortcut to the Home Screen
	Device_Search	0x3000 020E	Shortcut to the search function
	Device_Menu	0x3000 020F	Shortcut to the (application) menu
	Reserved	0x3000 0210	Reserved
		…	
		0x3000 02FF	
Function Keys	Function_Key_0	0x3000 0300	Soft and hard keys available on the VNC Client and Server user interface
	Function_Key_1	0x3000 0301	
	…	…	
	Function_Key_254	0x3000 03FE	
	Function_Key_255	0x3000 03FF	Reserved

Category	Mnemonic	KeySymValue	Description
Multimedia Keys	Multimedia_Play	0x3000 0400	Start media playing
	Multimedia_Pause	0x3000 0401	Pause media playing
	Multimedia_Stop	0x3000 0402	Stop media playing
	Multimedia_Forward	0x3000 0403	Forward
	Multimedia_Rewind	0x3000 0404	Rewind
	Multimedia_Next	0x3000 0405	Go to next track in playlist
	Multimedia_Previous	0x3000 0406	Go to previous track in playlist
	Multimedia_Mute	0x3000 0407	Mute the audio stream at source
	Multimedia_Unmute	0x3000 0408	Unmute the audio stream at source
	Multimedia_Photo	0x3000 0409	Take a photo
	Reserved	0x3000 040A	-
		…	
		0x3000 04FF	
Reserved	Reserved	0x3000 0500	-
		…	
		0x3000 FFFF	

Annex C (normative): Language Sets

C.1 Basic Set Latin-1

The following X11 key event values shall be supported for Latin-1.

- `'a' - 'z' (0x0061 – 0x007A)`
- `'A' - 'Z' (0x0041 – 0x005A)`
- `'0' – '9' (0x0030 – 0x0039)`

Latin-1 set shall be used, if a country specific set is not supported from both, the MirrorLink Server and Client.

Annex D (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
	Laurent Cremmer, RealVNC
	Dennis Fernahl, Carmeq (for Volkswagen AG)
	Jungwoo Kim, LG Electronics
	Mingoo Kim, LG Electronics
	Kari Kostiainen, Nokia Corporation
	Martin Lehner, Jambit
	Keun-Young Park, Nokia Corporation
	Tommy Park, LG Electronics
	Murali Soundararajan, Samsung
	Adrian Taylor, RealVNC
	Michael Wolf, Daimler AG
	Young-Rang Yoon, LG Electronics

History

Document history		
V1.3.0	October 2017	Publication
V1.3.1	October 2019	Publication