Car Connectivity Consortium

MirrorLink®

VNC based Display and Control

Version 1.1.7
(CCC-TS-010)

Version History

Version	Date	Comment
1.0.1	26 June 2011	Approved Version
1.1	31 March 2012	Approved Version
1.1.1	21 June 2012	Approved Errata Version
1.1.2	27 September 2012	Approved Errata Version
1.1.3	20 December 2012	Approved Errata Version
1.1.4	05 March 2013	Approved Errata Version
1.1.5	04 April 2013	Approved Errata Version
1.1.6	21 November 2013	Approved Errata Version
1.1.7	02 April 2015	Approved Errata Version

List of Contributors

Beckmann, Mark	Volkswagen AG
Benesch, Matthias	Daimler AG
Bose, Raja	Nokia Corporation
Brakensiek, Jörg (Editor)	Microsoft Corporation
Fernahl, Dennis	Carmeq (for Volkswagen AG)
Kim, Jungwoo	LG Electronics
Kim, Mingoo	LG Electronics
Kostiainen, Kari	Nokia Corporation
Lehner, Martin	Jambit
Park, Keun-Young	Nokia Corporation
Park, Tommy	LG Electronics
Soundararajan, Murali	Samsung
Taylor, Adrian	RealVNC
Wolf, Michael	Daimler AG
Yoon, Young-Rang	LG Electronics

LEGAL NOTICE

TABLE OF CONTENTS

VERSION HISTORY **2**
LIST OF CONTRIBUTORS **2**
LEGAL NOTICE **3**
TABLE OF CONTENTS **4**
LIST OF FIGURES **6**
LIST OF TABLES **7**
TERMS AND ABBREVIATIONS **9**
1 ABOUT **10**
2 INTRODUCTION **11**
3 MANAGING A VNC SESSION **12**
3.1 IDENTIFYING REMOTE APPLICATIONS AND THE VNC SERVER 12
3.2 LAUNCHING THE VNC SESSION 12
3.3 INTENTIONALLY TERMINATING THE VNC SESSION 12
3.4 UNINTENTIONALLY TERMINATING THE VNC SESSION 13
3.5 TESTING CONSIDERATIONS 13
4 TRADITIONAL VNC PROTOCOL PHASES **14**
4.1 HANDSHAKING PHASE 14
4.2 INITIALIZATION PHASE 15
4.3 FRAMEBUFFER UPDATE AND EVENT PHASE 16
5 VNC MIRRORLINK EXTENSION MESSAGES **20**
5.1 BYEBYE MESSAGE 22
5.2 DISPLAY CONFIGURATION MESSAGES 23
5.2.1 Framebuffer Scaling (VNC Server) *26*
5.2.2 Framebuffer Scaling (VNC Client) *28*
5.2.3 Handling of Different Framebuffer Aspect Ratios *29*
5.2.4 Handling of Server Pixel Aspect Ratios *30*
5.2.5 Handling of Display Orientation *30*
5.3 EVENT CONFIGURATION MESSAGES 32
5.4 EVENT MAPPING MESSAGES 36
5.5 KEY EVENT LISTING MESSAGE 38
5.6 VIRTUAL KEYBOARD TRIGGER MESSAGES 41
5.7 DEVICE STATUS MESSAGES 44
5.8 CONTENT ATTESTATION MESSAGES 48
5.9 FRAMEBUFFER BLOCKING NOTIFICATION 51
5.10 AUDIO BLOCKING NOTIFICATION 57
5.11 TOUCH EVENT 61
5.12 FRAMEBUFFER ALTERNATIVE TEXT 63
6 ADDITIONAL ENCODINGS AND PSEUDO ENCODINGS **64**
6.1 MIRRORLINK PSEUDO ENCODING 64
6.2 CONTEXT INFORMATION PSEUDO ENCODING 65
6.3 DESKTOP SIZE PSEUDO ENCODING 67
6.4 SCAN LINE BASED RUN-LENGTH ENCODING 68
6.5 TRANSFORM ENCODING 70
7 REFERENCES **73**
APPENDIX A – KNOB CONFIGURATION **75**
APPENDIX B – KEY EVENT MAPPING **76**

APPENDIX C – LANGUAGE SETS 79
BASIC SET LATIN-1 79

LIST OF FIGURES

Figure 1: MirrorLink VNC Setup 11
Figure 2: VNC Handshaking Phase 14
Figure 3: Initialization Phase 15
Figure 4: Framebuffer Update Phase 16
Figure 5: Server and Client Display Configuration 23
Figure 6: Framebuffer Scaling (VNC Server) – Message Flow 28
Figure 7: VNC Server Framing Option for non-scalable Clients 29
Figure 8: Server and Client Event Configuration 32
Figure 9: Example Event Mapping Message Flow 36
Figure 10: Key Event Listing Messages 38
Figure 11: Key Event Listing Updates 39
Figure 12: Example Virtual Keyboard Trigger Message Flow 41
Figure 13: Virtual Keyboard Trigger Text Entry Exchange Message Flow 43
Figure 14: Example Device Status Message Flow 44
Figure 15: Example Content Attestation Message Flow 48
Figure 16: Example Framebuffer Blocking Notification Message Flow 51
Figure 17: Decision Tree, blocking a Foreground Application 52
Figure 18: Decision Tree, blocking an Overlay of a Background Application 53
Figure 19: Framebuffer Blocking Notification Message Flow due to enabling Drive Mode 55
Figure 20: Example Audio Blocking Notification Message Flow 57
Figure 21: Decision Tree, blocking Audio 58
Figure 22: Analyzing the Audio Blocking Messages 60
Figure 23: Analyzing the Audio Packets 60
Figure 24: Touch Event Message Flow 61
Figure 25: Example Framebuffer Alternative Text Message Flow 63
Figure 26: Context Information (Example) 65
Figure 27: Example Transform Encoded Framebuffer Update 72
Figure 28: Coordinate System for Knob Events 75

LIST OF TABLES

Table 1: Requirements for Handshaking Phase Messages 14
Table 2: Requirements for Initialization Phase Messages 16
Table 3: Requirements for Framebuffer Update and Event Phase Messages 17
Table 4: VNC Extension Type Message Structure 20
Table 5: New Extension Types for MirrorLink Messages 21
Table 6: ByeBye Message 22
Table 7: Server Display Configuration Message 24
Table 8: Color Value Parameter (Big Endian) 24
Table 9: Client Display Configuration Message 25
Table 10: Framebuffer Downscaling Configurations 27
Table 11: Server Event Configuration Message 33
Table 12: Client Event Configuration Message 34
Table 13: Event Mapping Message 36
Table 14: Event Mapping Request Message 37
Table 15: Key Event Listing Message 39
Table 16: Key Event Listing Request Message 39
Table 17: Key Event Listing Options 40
Table 18: Virtual Keyboard Trigger Message 42
Table 19: Virtual Keyboard Trigger Request Message 42
Table 20: Device Status Message 45
Table 21: Device Status Request Message 46
Table 22: Content Attestation Response Message 49
Table 23: SignedInfo Element for HMAC-SHA-256 Signature Type 50
Table 24: Content Attestation Request Message 50
Table 25: Framebuffer Blocking Notification Message 55
Table 26: Audio Blocking Notification Message 60
Table 27: Touch Event Message 61
Table 28: Description of Individual Events 61
Table 29: Adaptation of Touch Event Pressure Values 62
Table 30: Framebuffer Alternative Text Message 63
Table 31: Framebuffer Alternative Text Request Message 63
Table 32: Additional VNC Encodings 64
Table 33: Context Information Pseudo Encoding 66
Table 34: Run-Length Encoding within a Framebuffer Update Message 68
Table 35: Run-Length Encoding for individual Line 68

Table 36: Run-Length Encoding for Individual Run 68
Table 37: Example Run-Length Encoding Values 69
Table 38: Transform Encoding within a Framebuffer Update Message 70
Table 39: Transform Encoding Structure 71
Table 40: Example Transform Encoded Framebuffer Update 72
Table 41: Knob Shift and Rotate Configuration Settings 75
Table 42: Head-Unit Knob Key Event Mapping 76
Table 43: Key Event Mapping 78

TERMS AND ABBREVIATIONS

A2DP	Bluetooth Advanced Audio Distribution Profile
ARP	Address Resolution Protocol
BT	Bluetooth
CDC	Communications Device Class; specified from USB Device Working Group
CE	Consumer Electronics; CE devices are referred to as mobile devices within this specification
DHCP	Dynamic Host Configuration Protocol
ECM	Ethernet Control Model; part of the CDC device class
HFP	Bluetooth Hands-free Profile
HSP	Bluetooth Headset Profile
HMI	Human Machine Interface
HU	Head-unit (this term is used interchangeably with the MirrorLink client)
HS	Head-set
IP	Internet Protocol
NCM	Network Control Model; part of the CDC device class
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.
RFB	Remote Framebuffer
RTP	Real-time Transport Protocol
TCP	Transmission Control Protocol
Touch Event	Touch events are used to describe touch screen action in which the user touches the screen with one or more separate fingers at different locations. Touch events are used to describe more complex touch action, like pinch-open or pinch-close.
UDP	User Datagram Protocol
UI	User Interface
UPnP	Universal Plug and Play
USB	Universal Serial Bus
VNC	Virtual Network Computing

MirrorLink is a registered trademark of Car Connectivity Consortium LLC

Bluetooth is a registered trademark of Bluetooth SIG Inc.

RFB and VNC are registered trademarks of RealVNC Ltd.

UPnP is a registered trademark of UPnP Forum.

Other names or abbreviations used in this document may be trademarks of their respective owners.

1 ABOUT

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

The document will focus on the interface functionality, its parameters and protocols only. It does not provide any guidelines for implementing the protocol. If there is a reference towards an implementation, this is of informative nature only.

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

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are following the notation as described in RFC 2119 [12].

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

2 INTRODUCTION

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 MAY include rotation or color 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.

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.

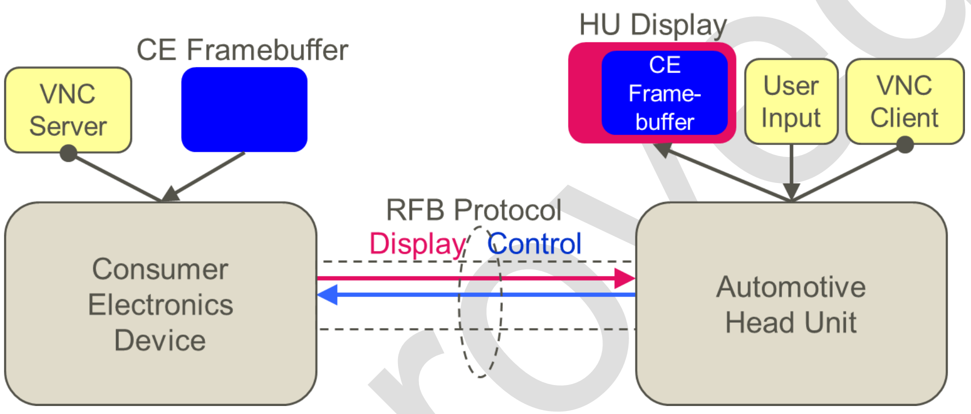

Figure 1: MirrorLink VNC Setup

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 signaled 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 behavior 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 client with only a few requirements, and a server having access to more processing capabilities. The protocol allows the 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 MUST implement the VNC Client functionality.

The MirrorLink Server MUST implement the VNC Server functionality

3 MANAGING A VNC SESSION

3.1 Identifying Remote Applications and the VNC Server

The identification of remote VNC based applications and of the VNC Server is described in [22].

3.2 Launching the VNC Session

The VNC Server start-up is automatically facilitated via UPnP. The MirrorLink Server's VNC Server MUST be running, when launching any application in response to a UPnP TmApplicationServer:1 service LaunchApplication action, as defined in [22]. The LaunchApplication action MUST 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 MUST be established, given the following steps:

1. VNC Server MUST listen for the VNC Client to make TCP connection at the provided URL.
2. VNC Client MUST make a TCP connection to the provided URL.
3. VNC Server and Client MUST initialize the VNC session according to the VNC protocol.

3.3 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 MUST 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 MUST terminate a VNC session using the VNC ByeBye message, as given below:

1. VNC Client MUST send a VNC ByeBye message. The VNC Client MUST 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 MUST respond with a VNC ByeBye message.
3. VNC Client MUST disconnect the TCP connection. The VNC Client SHOULD disconnect the TCP connection, if it does not receive a VNC ByeBye message back within 5s.
4. VNC Server SHOULD disconnect the TCP connection on detection of the client TCP disconnect or 5 s after sending the VNC ByeBye message, whatever comes first.

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

1. VNC Server MUST send a VNC ByeBye message. The VNC server MUST 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 MUST disconnect the TCP connection
3. VNC Server SHOULD disconnect the TCP connection on detection of the 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 client has previously launched it for the terminated session.

Terminating a VNC session MUST NOT impact the application status of any application on the MirrorLink server. If the MirrorLink Client decides to re-establish the VNC session, it MUST follow the steps given in Chapter 3.2.

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 MUST follow the steps given in Chapter 3.2.

In case the MirrorLink Client has intentionally terminated the VNC session, the MirrorLink Client MUST 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 MUST 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 MUST use a different URL than the previous VNC session.

3.4 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 MUST follow the steps given in Section 3.2.

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.

3.5 Testing Considerations

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

4 TRADITIONAL VNC PROTOCOL PHASES

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. Client to server and 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 section.

4.1 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.

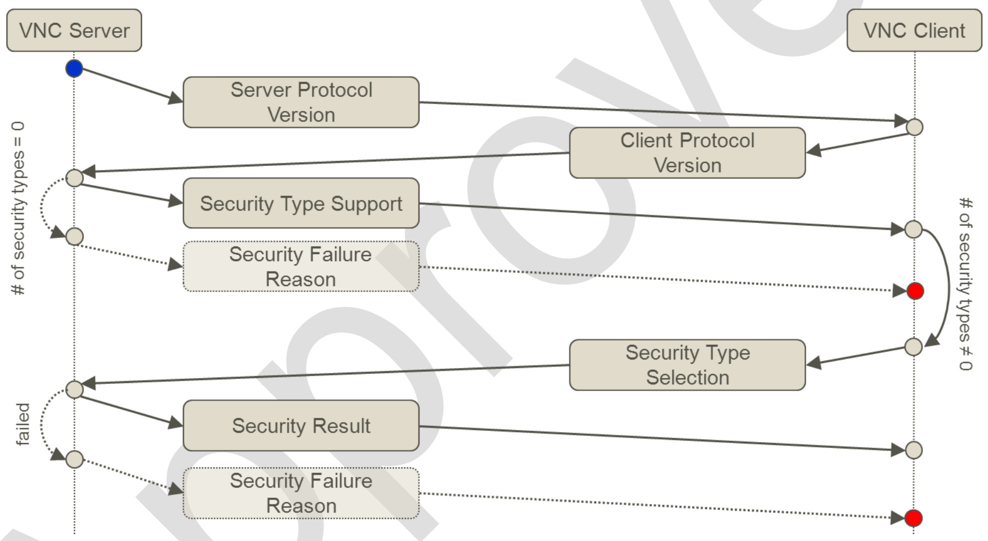

Figure 2: VNC Handshaking Phase

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

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)

Table 1: Requirements for Handshaking Phase Messages

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 MUST 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 client TCP disconnect or 5 s after sending the VNC Security Failure Reason message, whatever comes first.

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

4.2 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.

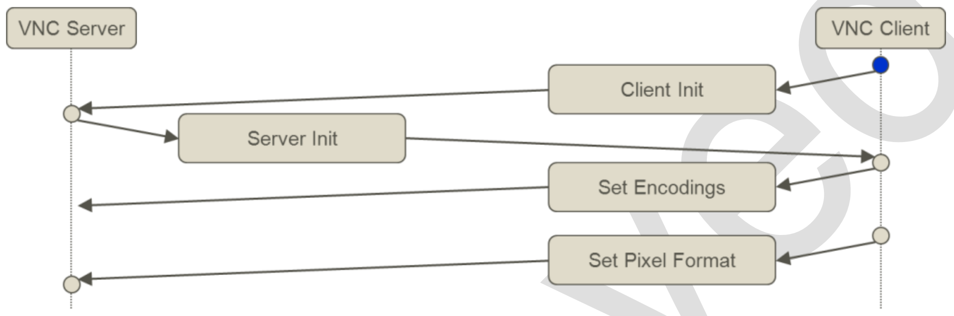

Figure 3: Initialization Phase

The Table 2 parameters MUST 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].

Message	Origin	Parameter	Mandatory Values
Client Init	Client	Shared-flag	(as specified in RFB)
Server Init (using native frame-buffer 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 Encod-ings is given in Table 32.
Set Pixel Format	Client	Bits-per-pixel	ARGB 888, RGB 565
		Depth	
		Big-endian-flag	
		True-colour-flag	
		Red-/Green-/Blue max	

Message	Origin	Parameter	Mandatory Values
		Red-/Green-/Blue shift	

Table 2: Requirements for Initialization Phase Messages

The MirrorLink Server MUST start in Landscape orientation. The MirrorLink Server MAY start in Portrait Mode within the VNC Server Init message, but it MUST switch to Landscape Mode using the Desktop Size Pseudo Encoding message, with the first Framebuffer Update message, following the MirrorLink Client's VNC Set Encoding message announcing MirrorLink support.

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

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

The VNC Client MUST initially send all supported encodings within a single Set Encodings message. The encoding order MAY be used from the VNC Server as an indication on the client's priority order (first entry has highest priority). Subsequent Set Encodings messages MUST 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 Set Encodings message MUST be sent prior the first Framebuffer Update Request message.

The VNC Client SHOULD NOT send a Set Pixel Format message, after the first Framebuffer Update Request message. Otherwise the VNC Client MUST always send only one Framebuffer Update Request at a time, and the VNC Client MUST send a new Set Pixel Format message, after a Framebuffer Update message has been received and before the next Framebuffer Update Request message is sent.

4.3 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.

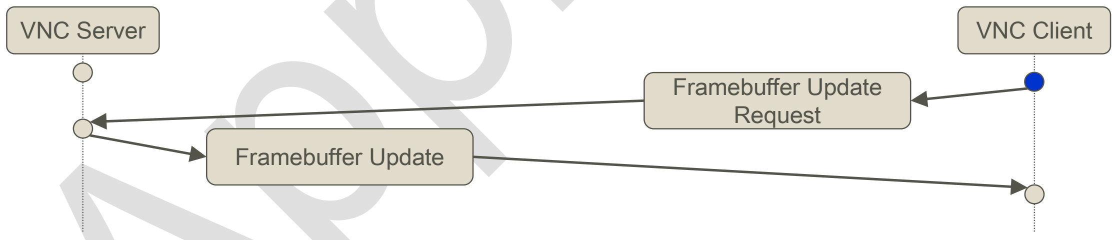

Figure 4: Framebuffer Update Phase

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

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	

Message	Origin	Parameter	Mandatory Values
		Width	
		Height	
		encoding-type	0 (Raw)
Set Colour Map Entries	Server	first color	(MUST 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	

Table 3: Requirements for Framebuffer Update and Event Phase Messages[1]

The VNC Client can request two types of framebuffer updates, using the incremental flag in the Framebuffer Update Request message.

- A '0' indicates the VNC Server MUST provide a non-incremental update, i.e. the VNC Server MUST provide the requested area, independent of whether it has changed or not (Exception: Within a Framebuffer Update message containing a Desktop Size Pseudo Encoding rectangle, the framebuffer data rectangle(s) SHOULD be skipped – see Section 6.3).
- 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 framebuffer update message, until the framebuffer content has actually changed. The VNC Server SHOULD NOT continuously send Framebuffer Update messages containing no framebuffer data. In any case, the VNC Client MUST support Framebuffer Update messages, where the provided framebuffer width or height equals zero.[2] (Exception: Within a Framebuffer Update message containing a Desktop Size Pseudo Encoding rectangle, the framebuffer data rectangles SHOULD be skipped – see Section 6.3).

The VNC Client SHOULD send the first Framebuffer Update Request message, after completion of the Display and Event Configuration, as specified in Chapters 5.2 and 5.2.5. The VNC Client MUST only request framebuffer data it can handle, without disconnecting the VNC session.

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

[2] Those framebuffer updates are not forbidden from the RFB specification specifically. Though some existing VNC clients, display warnings.

The VNC Server MUST NOT provide a Framebuffer Update outside the rectangular area, which has been requested from the VNC Client. The VNC Server MUST send a Framebuffer Update message only in response to a Framebuffer Update Request. The VNC Server MAY ignore Framebuffer Update Request messages, if multiple messages are sent at the same time. It is RECOMMENDED that the VNC Client sends only one Framebuffer Update Request 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 MUST 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, Appendix C defines language specific character sets, which MUST be supported.

The VNC Server MUST ignore all non-supported events. The VNC Server MUST 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 MUST follow behavior for supported X11 keys as given in keysymdef.h.

The VNC Server MUST 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 MUST ignore a key-release event, if no key-press event has been received before for the same key symbol value. The VNC Server MUST 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 behavior MUST 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 behavior.

For short key-press events, the VNC Client MUST 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 Key Event message, the VNC Client MUST map the Unicode/ISO 10646 key event range `U0100` to `U10FFFF` to the X11 key symbol value range `0x01000100` to `0x0110FFFF` as specified in [25]. For other Unicode values, found in ISO 10646 / Unicode, the following algorithm MUST be used: The X11 key symbol value is the character's Unicode number plus 0x01000000 [25].

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 Server Cut Text message, the VNC Client or server MUST 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 MUST be returned to the Latin-1 set within the same Client or Server Cut Text message. Latin-1 is the default character set.

5 VNC MIRRORLINK EXTENSION MESSAGES

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 section. 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 MUST use the MirrorLink message type and MUST follow the Table 4 fundamental design principle.

# 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

Table 4: VNC Extension Type Message Structure

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

The VNC Server and Client MUST 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 this specification, MUST 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.

Extension-Type	Message Name	Origin	Server Support	Client Support
0	ByeBye	Server	MUST	MUST
0	ByeBye	Client	MUST	MUST
1	Server Display Configuration	Server	MUST	MUST
2	Client Display Configuration	Client	MUST	MUST
3	Server Event Configuration	Server	MUST	MUST
4	Client Event Configuration	Client	MUST	MUST
5	Event Mapping	Server	MUST[3]	MAY[4]

[3] MirrorLink Server MUST NOT send message, if the Client has disabled its support

[4] MirrorLink Client MUST set the corresponding bit in the Client Event Configuration message

Extension-Type	Message Name	Origin	Server Support	Client Support
6	Event Mapping Request	Client	MUST[5]	MAY[6]
7	Key Event Listing	Server	SHOULD[3]	MAY[4]
8	Key Event Listing Request	Client	SHOULD[5]	MAY[6]
9	Virtual Keyboard Trigger	Server	SHOULD[3]	MAY[4]
10	Virtual Keyboard Trigger Request	Client	SHOULD[5]	MAY[6]
11	Device Status	Server	MUST	MUST
12	Device Status Request	Client	MUST	MUST
13	Content Attestation Response	Server	SHOULD	SHOULD
14	Content Attestation Request	Client	SHOULD	SHOULD
16	Framebuffer Blocking Notification	Client	MUST	MUST
18	Audio Blocking Notification	Client	MUST	MUST
20	Touch Event	Client	SHOULD[5]	SHOULD[4,6]
21	Framebuffer Alternative Text	Server	MAY	MAY
22	Framebuffer Alternative Text Request	Client	MAY	MAY

Table 5: New Extension Types for MirrorLink Messages

The VNC Client MUST disable the key event listing and the virtual keyboard trigger support in the Client Event Configuration 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 MUST use these extension messages, unless otherwise stated.

[5] MirrorLink Server MUST set the corresponding bit in the Server Event Configuration message

[6] MirrorLink Client MUST NOT send message, if the Server has disabled its support

5.1 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 MUST stop sending any further VNC message.

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

The ByeBye message is given in Table 6.

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

Table 6: ByeBye Message

The MirrorLink Server or Client MAY send a VNC ByeBye message in in lieu of a specific response message, which is otherwise mandated from this specification, 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 this specification. The receiving side then MAY 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 Set Encoding message, include the MirrorLink pseudo encoding.
- The MirrorLink Client's response to a Server Display Configuration message
- The MirrorLink Server's response to a Client Display Configuration message
- The MirrorLink Client's response to a Server Event Configuration message
- The MirrorLink Server's response to a Device Status Request message

5.2 Display Configuration Messages

The Server and Client Display Configuration message pair exchanges additional display information between the Client and the server. Based on the received information the Client MAY change the pixel format, sending a Set Pixel Format message. The server will use this information to optionally provide higher resolution virtual framebuffer copies. The message flow is shown in Figure 5.

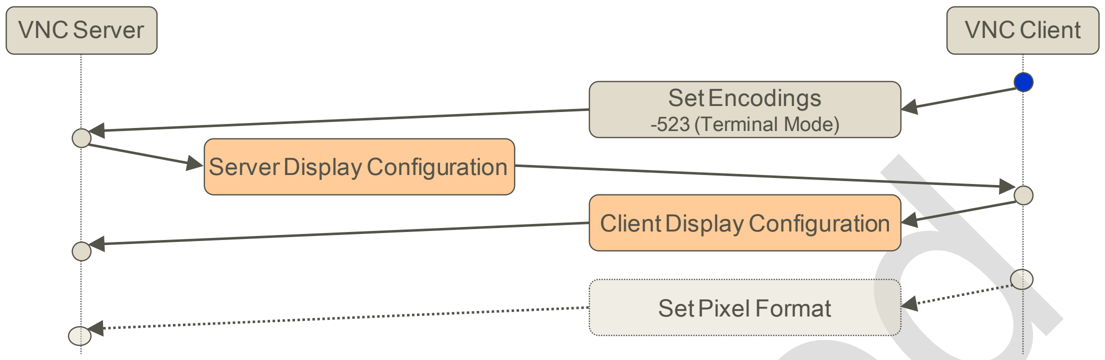

Figure 5: Server and Client Display Configuration

The Server Display Configuration message MUST be sent immediately in response to the Set Encoding message, indicating MirrorLink (-523) support, prior to any other message. The Client Display Configuration message MUST be sent immediately in response to the Server Display Configuration message, prior to any other message.

The Server Display Configuration message is given in Table 7. It will be responded to, by the VNC Client with, a Client Display Configuration 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 • The VNC Server MUST start in default orientation, given in the Server Init message.
		[1]	Server-side framebuffer rotation available • The VNC Server MUST start with no rotation.
		[2]	Server-side framebuffer up-scaling available
		[3]	Server-side framebuffer down-scaling available
		[5]	Server supports Framebuffer Alternative Text messages.
2	U16		Relative pixel width (set to zero, if relative width not known)
2	U16		Relative pixel height (set to zero, if relative height not known)
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
		[8]	24-bit RGB 888
		[15]	Any other 24-bit format
		[16]	16-bit RGB 565 (mandatory support for VNC Server)

# bytes	Type	Value	Description
		[17]	16-bit RGB 555 (15 bit color depth)
		[18]	16-bit RGB 444 (12 bit color depth)
		[19]	16-bit RGB 343 (10 bit color depth)
		[23]	Any other 16-bit format
		[24]	16-bit single color (grayscale) • Client MUST use *red_shift* and *red_mask* to set gray range
		[25]	8-bit single color (grayscale) • Client MUST use *red_shift* and *red_mask* to set gray range

Table 7: Server Display Configuration Message

The VNC Server MUST support ARGB 888 and RGB 565 color formats. It MAY support other formats as well. The VNC Client MUST select only color formats, supported from the server, using the Set Pixel Format message. With respect to this specification, the color encoding is represented in Table 8 (in case of big endian):

Parameter	ARGB 888	RGB 565	RGB 555	RGB 444	RGB 343	16-bit gray-scale	8-bit gray-scale
Bits-per-pixel	32	16	16	16	16	16	8
Color Depth	24	16	15	12	10	16	8
Red- max	255	31	31	15	7	0xFFFF	0x00FF
Green max	255	63	31	15	15	0	0
Blue max	255	31	31	15	7	0	0
Red shift	16	11	10	8	7	0	0
Green shift	8	5	5	4	3	0	0
Blue shift	0	0	0	0	0	0	0

Table 8: Color Value Parameter (Big Endian)

In case the bit-per-pixel is larger than the color depth, the unused bits SHOULD be set to Zero. It is RECOMMENDED to use color formats like RGB 444 and RGB 343 together with run-length encoding, in order to save transfer bandwidth.

The relative pixel width and height SHOULD be used to compensate for different pixel aspect ratio on client and server sides. The Server Display Configuration message SHOULD be sent only once. The VNC Client MUST ignore all subsequent Server Display Configuration messages.

The Client Display Configuration message is shown in Table 9.

# 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

# bytes	Type	Value	Description
			• If enabled, the VNC Client MUST use the Device Status Request message (section 5.7)
		[1]	Server-side framebuffer rotation used • If enabled, the VNC Client MUST use the Device Status Request message (section 5.7)
		[2]	Client-side framebuffer up-scaling available
		[3]	Client-side framebuffer down-scaling available
		[5]	Client supports Framebuffer Alternative Text messages
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]
4	U32	*Bit*	*Pixel Format Support (1 = support, 0 = not support)*
		[0]	32-bit ARGB 888
		[8]	24-bit RGB 888
		[16]	16-bit RGB 565
		[17]	16-bit RGB 555
		[18]	16-bit RGB 444
		[19]	16-bit RGB 343
		[24]	16-bit single color
		[25]	8-bit single color
4	U32	*Bit*	*Supported resize factors (1 = support, 0 = not support)*
		[0]	Resizing by factor of 1/1 (MUST BE ‘1’)
		[1]	Resizing by factor of 1/2
		[2]	Resizing by factor of 1/3
		[3]	Resizing by factor of 1/4
		[4]	Resizing by factor of 1/5
		[5]	Resizing by factor of 1/6
		[6]	Resizing by factor of 1/8
		[7]	Resizing by factor of 1/10
		[8]	Resizing by factor of 1/16
		[9]	Resizing by factor of 1/32
		[10]	Resizing by factor of 2/3
		[11]	Resizing by factor of 3/4

Table 9: Client Display Configuration Message

The MirrorLink Client MUST provide correct client display width and height values in resolution [pixel] and dimension [mm] within the Client Display Configuration message. The given values MUST accurately represent the display area on the MirrorLink Client fully dedicated to showing the MirrorLink Server’s framebuffer. The resolution [pixel] values MUST NOT be equal to zero (0). The dimension [mm] values MUST 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 MUST 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 MUST be 0, in which case a value of 900 mm MUST be used from the MirrorLink Server per default.

The MirrorLink Server MUST 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 Client Display Configuration message, it MUST be recognized as the Reference Display Resolution from the MirrorLink Server (i.e. 800 x 480).

The MirrorLink Client MUST provide a display, which would allow any CCC drive-certified application to be displayed, while the MirrorLink Client is in drive mode. The MirrorLink specification does not specify a minimum allowable size for the MirrorLink Client. Applications though, will be CCC certified to be rendered on a minimum resolution of 800 x 480 on a screen area of size 133.3 mm x 80 mm at a distance of 900 mm. It is within the responsibility of the MirrorLink Client to request a different resolution or display the received framebuffer on a different screen with a different distance. Applications may adapt to the provided screen details, but are not required to do so.

A MirrorLink Client MUST NOT have a framebuffer resolution smaller than 600x400.

A MirrorLink Client MUST NOT have a framebuffer size smaller than 105.75x70.5 mm (5” diagonal) at a distance of 900 mm. Note: The minimum size MUST be adjusted, in case the MirrorLink Client is known to be at a different distance, according to the following formulas:

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

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

The Client Display Configuration message MAY be sent again, to indicate changes to the client display configuration. The MirrorLink server MAY consider this for changing local scaling or virtual framebuffer setups, sending a Desktop Size Pseudo Encoding message to the MirrorLink Client.

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

The VNC Client SHOULD discard any framebuffer content in the Framebuffer Update message upon receiving the Desktop Size Pseudo-rectangle in that Framebuffer Update message and SHOULD ensure that the incremental flag is set to zero (false) in the next Framebuffer Update Request message.

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

The VNC Server MUST NOT use the provided information on the supported pixel formats from the MirrorLink Client to automatically change its pixel format, unless specifically allowed from the encoding (e.g. Transform encoding).

5.2.1 Framebuffer Scaling (VNC Server)

The MirrorLink VNC Server MUST down-scale its framebuffer to fit into the framebuffer resolution provided from the MirrorLink Client in the Client Display Configuration message, if the MirrorLink Client indicates no support for down-scaling and if the MirrorLink Server’s framebuffer resolution exceeds[7] a resolution of 800x480, as advertised in the VNC Server Init message.

[7] A framebuffer resolution exceeds 800x480 if its width is larger than 800 pixels or its height is larger than 480 pixels.

The MirrorLink VNC Server MUST follow the framebuffer down-scaling configurations, as given in the Table 10.

Down-scaled framebuffer received from MirrorLink Client			MirrorLink Server Framebuffer FB_{Server}		
			FB_{Server} ≤ 800x480	800x480 < FB_{Server} ≤ 1024x600	FB_{Server} > 1024x600
MirrorLink Client Framebuffer FBCli-	FB_{Client} ≤ 800x480		FB_{Client} FB_{Server}	FB_{Client} 800x480	FB_{Client} 800x480
	800x480 < FB_{Client} ≤ 1024x600	Down-scaling support	N/A	FB_{Client} FB_{Server} 800x480	FB_{Client} 1024x600 800x480
		No Down-scaling support	N/A	FB_{Client} 800x480	FB_{Client} 800x480
	FB_{Client} > 1024x600		N/A	N/A	FB_{Client} 1024x600 800x480

Table 10: Framebuffer Downscaling Configurations

The MirrorLink VNC Server has one of the following three options, dependent on the MirrorLink Client and Server properties:

1. Scale to the Client framebuffer resolution (FB_{Client}),
2. Do not scale (FB_{Server}) or
3. Scale to a fixed resolution

The MirrorLink VNC Server MUST NOT scale, if the client display width or height [pixel] within the Client Display Configuration message is unknown.

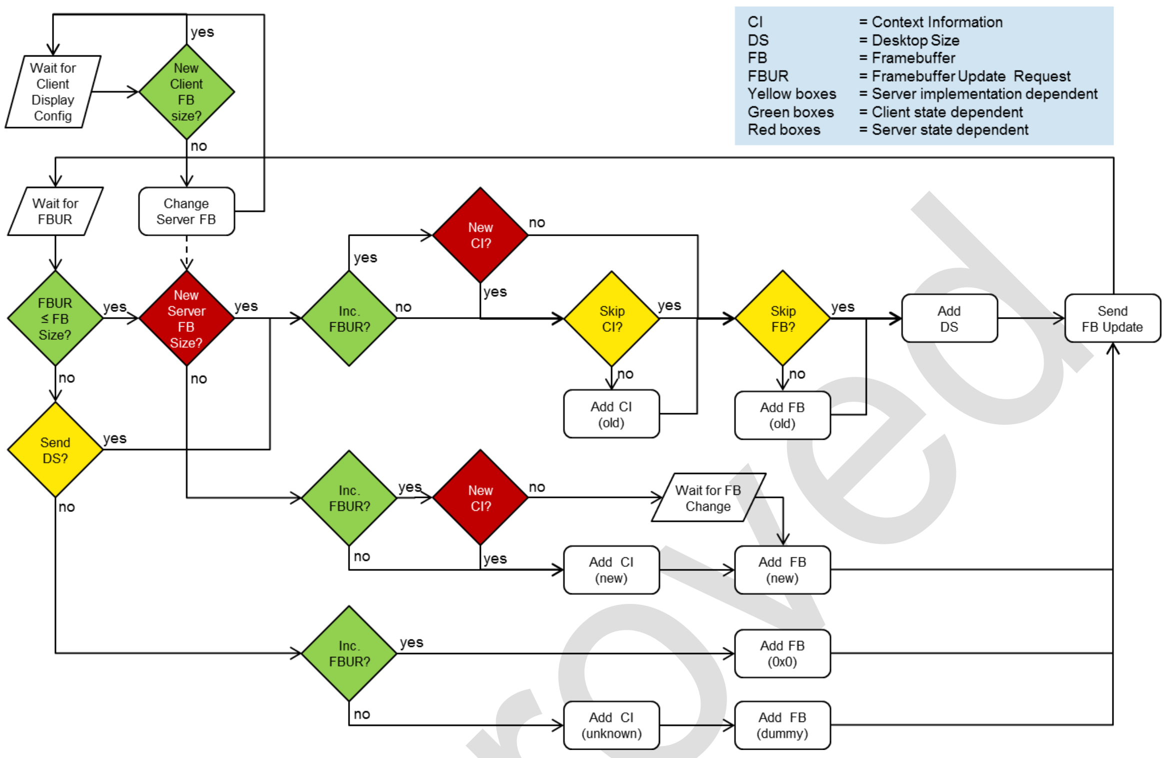

Figure 6: Framebuffer Scaling (VNC Server) – Message Flow

If required, the MirrorLink VNC Server MUST down-scale on receiving a Client Display Configuration message and the MirrorLink VNC Server MUST indicate that it started to downscale in the 1st Framebuffer Update message using Desktop Size Pseudo Encoding rectangle (Note: The framebuffer data, which may be included in that Framebuffer Update is not down-scaled to the new resolution yet).

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

The MirrorLink Server SHOULD down-scale in order to avoid unnecessary network traffic in case the MirrorLink Client supports down-scaling.

The MirrorLink Server MUST preserve the framebuffer's aspect ratio during scaling and MUST NOT upscale to a resolution larger than given in the latest Client Display Configuration message.

5.2.2 Framebuffer Scaling (VNC Client)

The MirrorLink Client MUST down-scale from within[8] 800x480 to the framebuffer resolution provided in Client Display Configuration message or it MUST follow the MirrorLink Server's framebuffer resolution in the VNC Server Init message. In latter case, the MirrorLink Client MUST always request a framebuffer resolution within the MirrorLink Server's resolution.

The MirrorLink Client MUST support up-scaling to the framebuffer resolution provided in Client Display Configuration message, if that resolution exceeds 800x480.

[8] A framebuffer resolution is within 800x480, if its width is equal to or smaller than 800 pixels and its height is equal to or smaller than 480 pixels.

The MirrorLink Client MUST provide valid client display width and height [pixel] within the Client Display Configuration message.

The MirrorLink Client MAY request a smaller framebuffer resolution from the MirrorLink Server (sending a Client Display Configuration message), and then perform internal up-scaling. This is transparent to the MirrorLink Server.

If both VNC Server and VNC Client support scaling, it is up to the VNC Server to decide whether to scale (either in part of full) or not to scale.

The VNC Client MUST frame the server's framebuffer, if it is smaller and if the VNC Client cannot upscale on its own, as shown in Figure 7. It is up to the VNC Client to decide where the framebuffer is positioned in the frame.

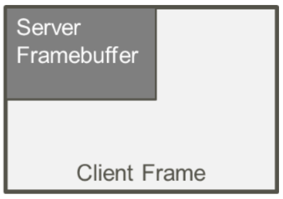

Figure 7: VNC Server Framing Option for non-scalable Clients

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

5.2.3 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 MUST 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 MUST NOT exceed the MirrorLink Client's framebuffer resolution, as given in the Client Display Configuration message. The MirrorLink Server MUST use the padded resolution within any VNC Server Init and Desktop Size Pseudo Encoding message.

Example 1: A MirrorLink Server with a Display resolution of 1280x720 pixel (FAR = 16:9) is providing its framebuffer to a MirrorLink Client with a Display Resolution of 800x480 (FAR = 5:3). The MirrorLink Server will need to downscale its framebuffer to 800x450 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 1280x720 pixel (FAR = 16:9) is providing its framebuffer to a MirrorLink Client with a Display Resolution of 1024x576 (FAR = 16:9). The MirrorLink Server will need to downscale its framebuffer to 1024x576 to be within the Client framebuffer resolution. The Server MUST NOT add any pad rows or columns to its framebuffer.

The VNC Server MUST use a framebuffer resolution in the VNC Server Init and Desktop Size Pseudo Encoding message, which results in a framebuffer aspect ratio (in its landscape configuration) of at least 1.33 (4:3) and not exceeding 2.00 (2:1), unless one of the following two exceptions is fulfilled:

1. The VNC Server MAY provide an extended aspect ratio, if it matches the VNC Client's framebuffer aspect ratio, as provided within the VNC Client Display Configuration message, and if all applications running in the foreground are supporting the same and their certification status extends to the extended aspect ratio, when being displayed on the extended aspect ratio.

2. The VNC Server MAY provide an extended aspect ratio, if all applications running in the foreground are supporting the same aspect ratio and their certification status extends to the extended aspect ratio, when being displayed with an 15:9 aspect ratio (reference display).

5.2.4 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)[9].

The MirrorLink Server MUST compensate for a Server PAR not equal to 1 (one), prior to transmitting its framebuffer to the MirrorLink Client. The MirrorLink Server MUST use the PAR-compensated resolution within any VNC Server Init and Desktop Size Pseudo Encoding message. The MirrorLink Server MUST set the relative pixel width and height values in the Server Display Configuration message to 1.

Example 1: A MirrorLink Server with a Display resolution of 1280x720 pixel and a Display size of 4:3, has a pixel aspect ratio of (4/1280) / (3/720) = 3/4. The Server will need to re-scale its frame-buffer to 960x720 to achieve a PAR of 1.

Example 2: A MirrorLink Server with a Display resolution of 1280x720 pixel and a Display size of 16:9, has a pixel aspect ratio of (16/1280) / (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 MUST adjust the framebuffer resolution within the Client Display Configuration message to fully match the PAR compensation (i.e. the framebuffer resolution describes squared pixels). The reported physical dimensions MUST NOT be adjusted in this case.

5.2.5 Handling of Display Orientation

The MirrorLink Server can inform the MirrorLink Client in its Server Display Configuration message, whether can change its framebuffer orientation in the "`Framebuffer Orientation Switch`" flag. In response, the MirrorLink Client has a similar flag in its Client Display Configuration message to inform the MirrorLink Server, whether it will use the VNC Device Status Request message, to request a framebuffer orientation change.

If both the MirrorLink Client and the MirrorLink Server have enabled the "`Framebuffer Orientation Switch`" flag, in their respective Client and Display Configuration messages, the following requirements are valid:

- In case the application launches within the current orientation
    - MirrorLink Server moves the application into the foreground.

- In case the application launches within the different orientation,
    - MirrorLink Server moves the application into the foreground.
    - MirrorLink Server MUST send a Desktop Size Pseudo Encoding message
    - MirrorLink Server MUST send a Device Status message with the new orientation
    - MirrorLink Client MAY prohibit new orientation, by following any one of the approaches
        - Send a Device Status Request message with the new orientation or
        - Send a Framebuffer Blocking Notification message with "`UI Layout Not Supported`".
    - On receiving the prohibit information the MirrorLink Server MUST
        - Inform the application to change the orientation.

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

        - Move the application into the background, if the application does not support the new orientation.

If the MirrorLink Client and/or the MirrorLink Server have disabled the "`Framebuffer Orientation Switch`" flag, in their respective Client and Display Configuration messages, the following requirements are valid:

- In case the application launches in Landscape orientation
    - MirrorLink Server moves the application to the foreground.
- In case the application launches in Portrait orientation
    - MirrorLink Server requests the application to change to Landscape orientation.
    - If application does not support Landscape, the MirrorLink Server moves the application to the background or terminates the application.

The MirrorLink Server MUST NOT change its framebuffer rotation, unless explicitly requested from the MirrorLink Client.

5.3 Event Configuration Messages

The Server and Client Event Configuration message pair provides additional information about event handling, i.e. which key and pointer events are natively supported on the server and client. This information helps the server to map specific incoming client events to server events.

The message flow is shown in Figure 8.

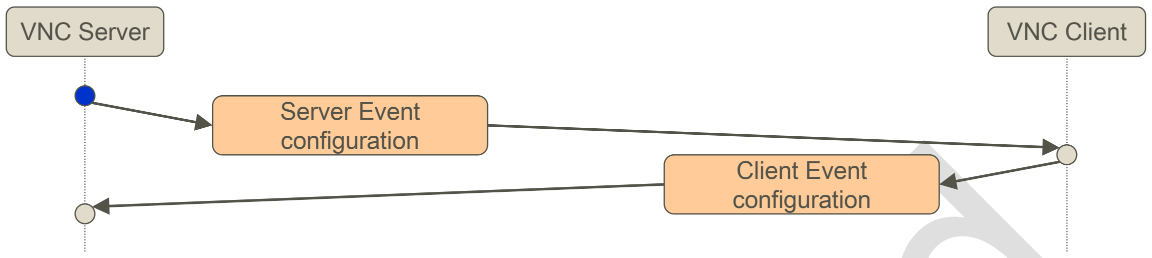

Figure 8: Server and Client Event Configuration

The Server and Client Event Configuration messages have to follow the rules, given next.

- The MirrorLink server MUST provide all key events, it natively supports. This MAY include key events known to be only supported from specific applications, e.g. the Device_Zoom_in event, being supported from a Navigation application.
- Support for cursor keys MUST be indicated from the MirrorLink server in the Knob key section. A trackball push event MAY be indicated as a Knob_2D_n_shift_push key event.
- The MirrorLink client MUST provide all key events it natively supports. This MAY include soft buttons, e.g. rendered on a touch screen.
- The MirrorLink client MUST be able to utilize all key events, advertised in the Client Event Configuration message.
- The MirrorLink server MUST NOT assume that the MirrorLink client will support all key events, supported from the server.
- The MirrorLink server MUST use Event Mapping messages, described in the next paragraph, to remap any missing vital key event, if the client supports those messages. Vital in this case describes key events, mandatory for operating the basic MirrorLink server device functionality. It is the MirrorLink server's responsibility to ensure basic operation, within the limits given from the MirrorLink client event support.

The Server Event Configuration message MUST be sent immediately in response to the Set Encoding message, indicating MirrorLink (-523) support, but after the Server Display Configuration has been sent. The message MAY be delayed only until reception of the Client Display Configuration message. The Client Event Configuration message MUST be sent immediately in response to the Server Event Configuration message, prior any other message.

The Server Event Configuration message is given in Table 11.

# 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)
2	U16		Keyboard layout – Country code (according ISO 3166-1 alpha-2)
2	U16		UI Language – Language code (according ISO 639-1)

# bytes	Type	Value	Description
2	U16		UI Language – Country code (according ISO 3166-1 alpha-2)
4	U32	*Bit l*	*Knob keys* (Bit mask according Table 41) • '1': Server supports knob key events • '0': Server does not support knob key events
4	U32	*Bit m*	*Device keys* (Bit mask according Table 43) • '1': Server supports device key events • '0': Server does not support the key events
4	U32	*Bit n*	*Multimedia keys* (Bit mask according Table 43) • '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 (T9) events ('0', … ,'9', '#', '*")
		[1]	Virtual keyboard trigger support
		[2]	Key event listing support
		[3]	Event mapping support (MUST be '1')
		[15:8]	Number of Function keys, the 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) Must 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. Must be 0x00, if the VNC Server does not support touch events.
		[31:24]	Touch event pressure mask, supported from the VNC Server. Must be 0x00, if the VNC Server does not support touch events.

Table 11: Server Event Configuration Message

If the VNC Server supports touch events, the pressure masks 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 MUST NOT be used if the "Touch events" bit is enabled.

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

The payload structure of the Client Event Configuration message is fully symmetrical with the Server Event Configuration message, as shown in Table 12.

# bytes	Type	Value	Description
1	U8	128	Message-type

# bytes	Type	Value	Description
1	U8	4	Extension-type
2	U16	28	Payload length
2	U16		Keyboard layout – Language code (according ISO 639-1)
2	U16		Keyboard layout – Country code (according ISO 3166-1 alpha-2)
2	U16		UI Language – Language code (according ISO 639-1)
2	U16		UI Language – Country code (according ISO 3166-1 alpha-2)
4	U32	*Bit l*	*Knob keys* (Bit mask according Table 41) • '1': Client supports knob key events • '0': Client does not support knob key events
4	U32	*Bit m*	*Device keys* (Bit mask according Table 43) • '1': Client supports device key events • '0': Client does not support device key events
4	U32	*Bit n*	*Multimedia keys* (Bit mask according Table 43) • '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 (T9) events ('0', … ,'9', '#', '*")
		[1]	Virtual keyboard trigger support
		[2]	Key event listing support
		[3]	Event Mapping support
		[15:8]	Number of Function keys, the 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) Must 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. Must be 0x00, if the VNC Client does not support touch events.
		[31:24]	Touch event pressure mask, supported from the VNC Client. Must be 0x00, if the VNC Client does not support touch events.

Table 12: Client Event Configuration Message

A MirrorLink Client, not supporting Pointer events, MUST support the Knob_2D_0_shift_push event and the knob event pairs listed below and MUST enable them in the Client Event Configuration 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

A MirrorLink Server MUST pass the following knob events to the applications and MUST enable them in the Server Event Configuration 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

A MirrorLink Server MUST 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 MUST 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 MUST follow the same ruling as the server side pressure mask.

A MirrorLink Server MUST support Pointer events, at least to navigate and select user input elements, and MUST enable it in the Server Event Configuration message.

The MirrorLink Client SHOULD follow the language settings of the MirrorLink Server.

5.4 Event Mapping Messages

The Event Mapping and Event Mapping Request message pair provide the client with information about the server mapping of high key symbol values, as listed in Appendix B. The client can send an Event Mapping Request message at any time, either requesting or setting specific mapping within the server. The server MUST always send an Event Mapping message in response to an Event Mapping Request message. If the server changes any event mapping locally, the server MUST inform the client via an Event Mapping message.

The VNC Server SHOULD allow remapping of any key event defined in Appendix B.

The VNC Server SHOULD change the mapping of the device and multimedia keys only, if the following rule applies.

- The key event in question does not fundamentally change the key event context. I.e. mapping Device_Delete to Device_Clear MAY be done, whereas mapping Device_Zoom_In to Multimedia_Play SHOULD NOT be done.

For clarification: If the MirrorLink Client has mapped the client key event A to the server key event B, the MirrorLink Client MUST continue sending key event A, rather than key event B. The key event B MAY or MAY NOT be a key event as listed in Appendix B. Note: In case of an infinite mapping loop, the behavior is implementation dependent.

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

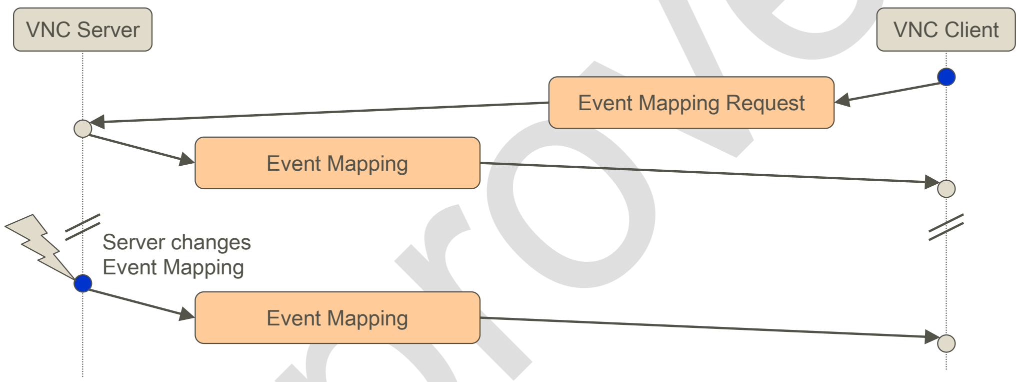

Figure 9: Example Event Mapping Message Flow

The server MUST respond to any Event Mapping Request message immediately.

The MirrorLink server SHOULD send Event Mapping messages, immediately following the Client Event Configuration message, in order to remap any missing key event functionality.

The Event Mapping message is given in Table 13.

# 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 (0 = client key value not mapped from server)

Table 13: Event Mapping Message

The Default Mapping Request message is given in Table 14.

# bytes	Type	Value	Description
1	U8	128	Message-type

# bytes	Type	Value	Description
1	U8	6	Extension-type
2	U16	8	Payload length
4	U32		Client Key Symbol Value
4	U32		Server Key Symbol Value (0 = request value from server)

Table 14: Event Mapping Request Message

If the server locally changes any event mapping it MUST send an Event Mapping message with appropriate client and server key symbol values, unless the MirrorLink client has requested no such messages in the Client Event Configuration message. The VNC Server SHOULD NOT send Event Mapping messages for client events not supported from the VNC Client. The VNC Client MUST ignore all Event Mapping messages for client events not supported.

If the server does not support a new mapping request according to the Event Mapping Request message, it MUST send an Event Mapping message, containing the existing mapping. The server key symbol value is set to zero if the event is not mapped.

To summarize this, the MirrorLink Server has to respond to any Event Mapping Request message with an Event Mapping message. The Client Key Symbol Value in the response message is identical to the value in the request message. The Server Key Symbol Value follows the rules given below:

- If the Server does not support the key event, the returned value is 0x00000000.
- If the Server can remap the key event to the requested value, that value is returned.
- If the Server cannot remap the key event, the existing mapping is returned.

5.5 Key Event Listing Message

The Key Event Listing and Key Event Listing Request message pair provides the client with information about the next valid characters, while entering text information. The VNC Server support is announced in the Server Event Configuration message. The VNC Server MUST provide a default key event list, which MUST be used from the VNC Client in the following cases:

1. Use as a reference for any first incremental key event update. I.e. white listed events are added to the default list, and black listed events are removed from the default list.
2. Use as the key event list, if no key event list updates have been received.

The client MAY start the key event listing at any time. In case a text entry is REQUIRED, the server will provide the initial list of valid keys, which gets updated after each key event, either as incremental or full update. An example message flow is shown in Figure 10.

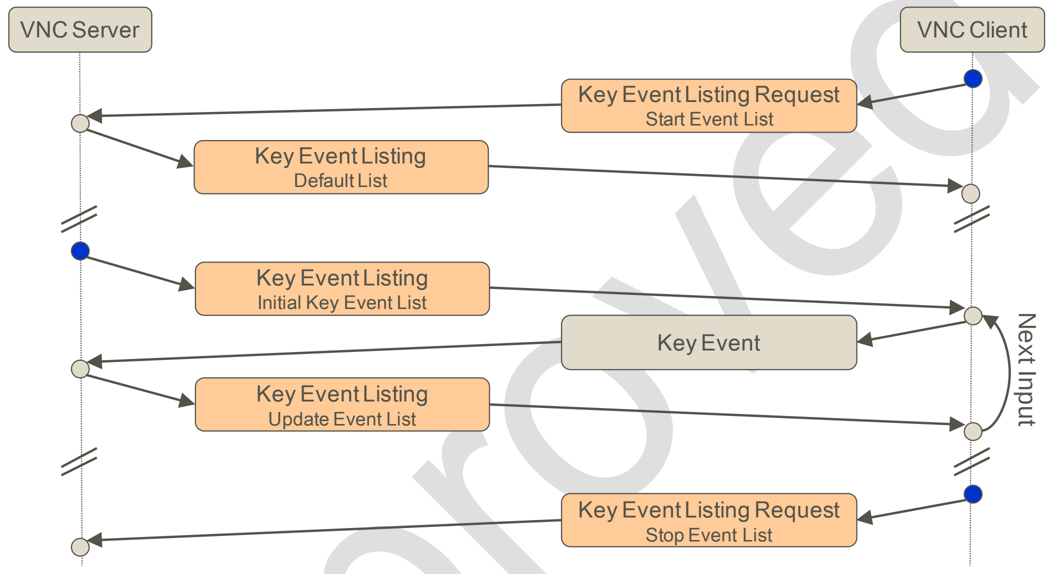

Figure 10: Key Event Listing Messages

The VNC Server MUST send Key Event Listing messages only, if the VNC Client has enabled or requested them. The VNC Client SHOULD NOT assume that the VNC Server will send Key Event Listing messages even if it has indicated support (the current application MAY NOT be able to support this feature). If the client does not receive Key Event Listing messages it MUST use the default key event list instead.

The Key Event Listing message is given in Table 15.

# bytes	Type	Value	Description
1	U8	128	Message-type
1	U8	7	Extension-type
2	U16	4 + 4*n	Payload length
1	U8	*Bit*	*Configuration*
		[0]	Incremental flag (0 = non-incremental, 1 = incremental)
		[1]	Listing flag (0 = black list, 1 = white list)
		[2]	Default key event list flag • '1' – the key event list contains the default key event list. • '0' – the list contains a key event list update, either with respect to the previous update (incremental flag = 1) or with respect to the default list (incremental flag = 0).

# bytes	Type	Value	Description
		[3]	Key event list follows flag (0 = last list, 1 = event list follows)
1	U8	*n*	# key events in list
2	U16		Key event counter
4 * *n*	U32 array		KeySymValue list used to define the next valid character

Table 15: Key Event Listing Message

During the incremental key event list update, the white list contains all key symbols to be added to the key event list, whereas black lists contains all key symbols to be removed from the key event list. A key event list update can be separated into individual Key Event Listing messages, immediately following each other, using the Key Event List Follows flag. This allows mixing of black and white listing updates. The Key Event Listing updates flow is shown in Figure 11.

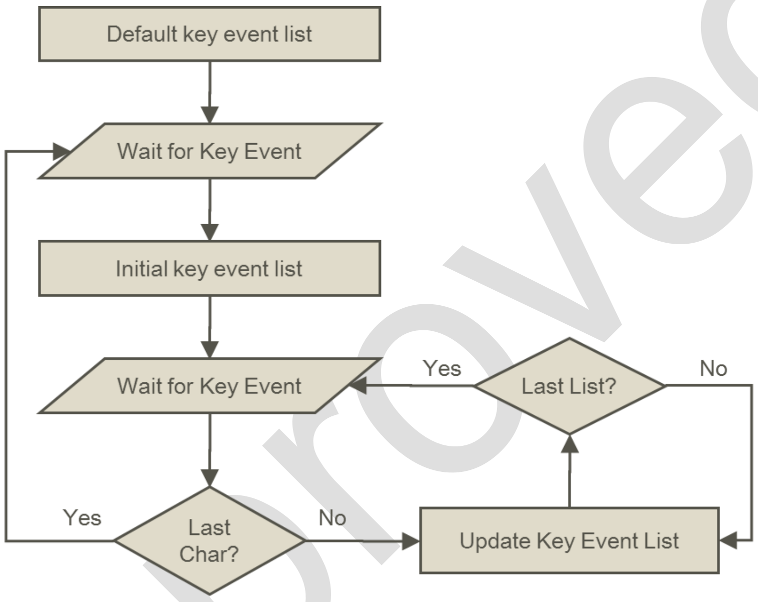

Figure 11: Key Event Listing Updates

The valid key symbol list MUST NOT differentiate between upper and lower case characters

The Key Event Listing Request message is shown in Table 16.

# bytes	Type	Value	Description
1	U8	128	Message-type
1	U8	8	Extension-type
2	U16	4	Payload length
4	U32	*Bit*	*Configuration (0 = Disable, 1 = Enable)*
		[0]	Server key event listing ‘1’: Start event listing ‘0’: Stop event listing
		[1]	Incremental updates supported from MirrorLink client
		[2]	Reset key event counter

Table 16: Key Event Listing Request Message

The key event counter SHOULD be used to synchronize the server key event listing message to key events. The counter value MUST represent all received key events (key press events only). The counter MUST be set to zero on the reception of the Client Event Configuration message or if the specific bit is set in the Key Event Listing Request message. The counter MUST roll over to zero, once the maximum number is reached.

If the next key event is entered faster than the VNC Server provides a key event list update, the VNC Client MUST use the default key event list instead.

Table 17 clarifies the different combination options with the Key Event Listing message.

Incremental Flag	Listing Flag	Default Key Event Flag	Description of expected behavior
Incremental	White List	List Update	Adds key symbols to the current list
Non-Incremental			Updates current list, using an empty list as starting point
Incremental	Black List		Removes key symbols from current list
Non-Incremental			Updates current list, using default list as starting point
Incremental	White List	Default List	Updates default list, using previous default list as starting point
Non-Incremental			Updates default list, uses an empty list as starting point
Incremental	Black List		Updates default list, using previous default list as starting point
Non-Incremental			Not allowed – Behavior is implementation specific

Table 17: Key Event Listing Options

The following examples will explain the principle of the Key Event listing from a MirrorLink Client perspective, in case multiple Key Event Listing messages are received for the same key event counter value (the default list in all examples is [A, B]):

Example 1:

```
1. KeyEventList(inc., white-list, key-event-list-follows) = [C, D]
2. KeyEventList(inc., white-list, last-list) = [E, F]
```

The resulting key event list is [A, B, C, D, E, F] – the second list adds to the first one

Example 2:

```
1. KeyEventList(inc., white-list, key-event-list-follows) = [C, D]
2. KeyEventList(inc., black-list, last-list) = [B, C]
```

The resulting key event list is [A, D] – first list adds, second removes key events

Example 3:

```
1. KeyEventList(inc., white-list, last-list) = [C, D]
2. KeyEventList(inc., white-list, last-list) = [E, F]
```

The resulting key event list is [A, B, C, D, E, F] – the second list add to the first one

Example 4:

```
1. KeyEventList(non-inc., white-list, last-list) = [C, D]
2. KeyEventList(non-inc., white-list, last-list) = [E, F]
```

The resulting key event list is [E, F] – the second list replaces the first one

5.6 Virtual Keyboard Trigger Messages

The Virtual Keyboard Trigger and Virtual Keyboard Trigger Request message pair informs the client that a text input is expected. The client can use this information, e.g. to bring up a virtual keyboard. In addition the message provides, if available, information about the current cursor position and the text entry box. The client can start and stop this service at any time.

The message flow is given in Figure 12.

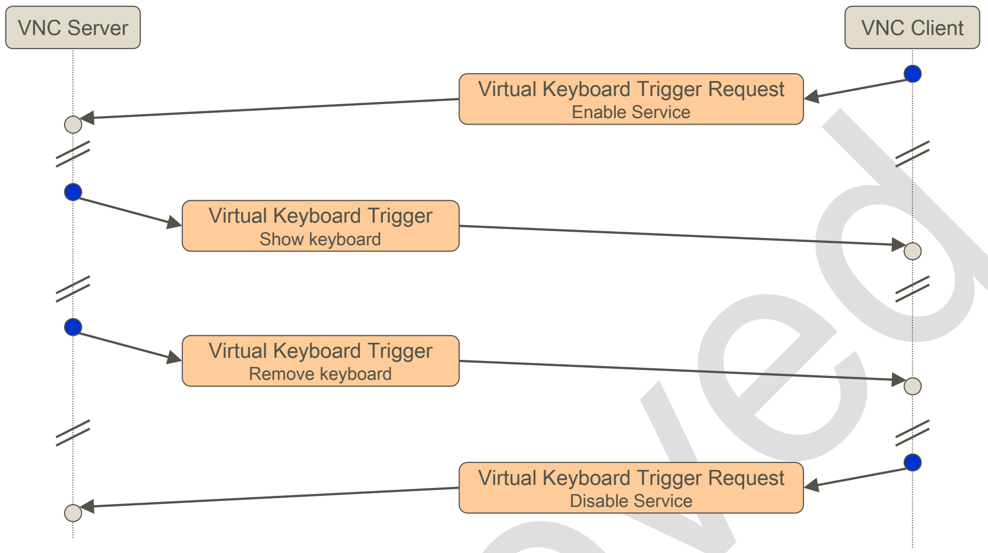

Figure 12: Example Virtual Keyboard Trigger Message Flow

The server MUST send Virtual Keyboard Trigger messages only, if it has set the "Virtual keyboard trigger support" flag in the Server Event Configuration message and the client has set the "Virtual keyboard trigger support" flag in the Client Event Configuration message.

The Virtual Keyboard Trigger message is given in Table 18.

# bytes	Type	Value	Description
1	U8	128	Message-type
1	U8	9	Extension-type
2	U16	16	Payload length
4	U32	*Bit*	*Configuration (0 = no, 1 = yes)*
		[0]	Valid cursor position
		[1]	Valid text input area
		[2]	Key Event listing follows
		[3]	Virtual keyboard control (0 = show keyboard, 1 = remove keyboard)
		[4]	Text Entry Exchange (0 = not available, 1 = available)
		[15:8]	Virtual Keyboard Type: • 0x00: Unknown • 0x01: QWERTY keyboard • 0x02: Numeric keyboard (including '+', and '#')
2	U16		Cursor – X Position
2	U16		Cursor – Y Position
2	U16		Text input area – X-Position

# bytes	Type	Value	Description
2	U16		Text input area – Y-Position
2	U16		Text input area – Width
2	U16		Text input area – Height

Table 18: Virtual Keyboard Trigger Message

The Virtual Keyboard Trigger Request message is given in Table 19.

# bytes	Type	Value	Description
1	U8	128	Message-type
1	U8	10	Extension-type
2	U16	4	Payload length
4	U32	*Bit*	*Configuration (0 = no, 1 = yes)*
		[0]	Enable trigger
		[1]	Support text entry exchange
		[15:8]	Maximum number of characters supported as initial text. A zero value (0) indicates no length limitation.

Table 19: Virtual Keyboard Trigger Request Message

The MirrorLink Server SHOULD provide a virtual keyboard. The MirrorLink Server MAY provide other different keyboard types (e.g. alphanumeric, numeric). While in drive mode, the MirrorLink Server MUST provide the same platform-specific virtual keyboard as used in Drive-level certification. Any MirrorLink-Certified or Member-Certified Application MAY provide its own virtual keyboard.

The MirrorLink Client MAY provide a virtual keyboard. In that case, the MirrorLink Client MUST NOT overlay its own virtual keyboard, while showing a MirrorLink-Certified or Member-Certified Application, unless the MirrorLink Server and Client are supporting a Client Virtual Keyboard and the MirrorLink-Certified or Member-Certified Application is requesting it explicitly.

The MirrorLink Client MAY send key events, while the MirrorLink Server shows a virtual keyboard.

The MirrorLink Server MUST support Unicode in key events and Client Cut Text messages.

If the VNC client and VNC server both support text entry exchange, the VNC server is able to provide an initial text entry to the VNC client, to be used for virtual keyboard operations. In addition the client will provide the entered text back to the server within a single message, rather than individual key events.

The text entry protocol extension is shown in Figure 13.

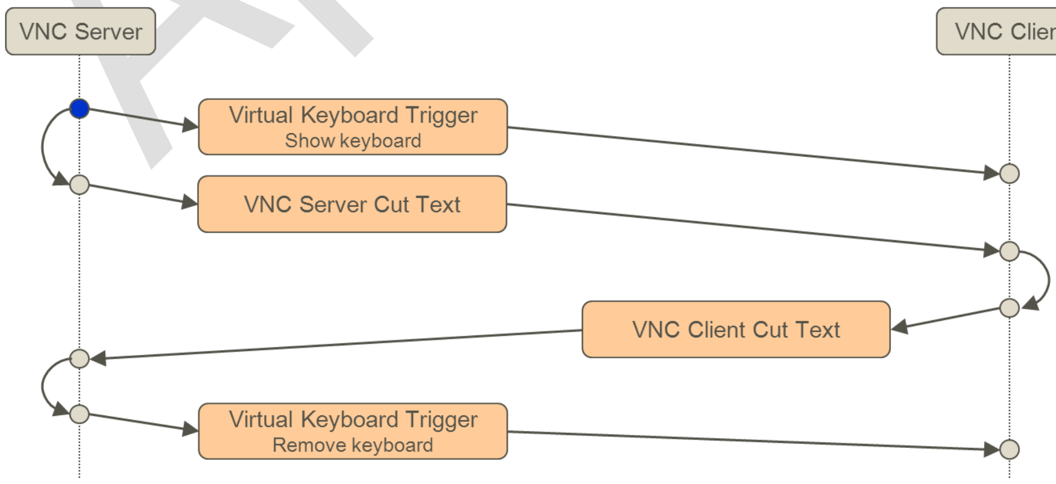

Figure 13: Virtual Keyboard Trigger Text Entry Exchange Message Flow

If both the VNC client and VNC server support text entry exchange, the following protocol MUST be implemented:

1. The VNC server MUST send a Virtual Keyboard Trigger message to show the keyboard with the Text Entry Exchange bit MUST be set to ‘1’. The Virtual Keyboard Type entry SHOULD give the client an indication on the requested keyboard type.
2. The VNC server MUST send a VNC Server Cut Text message, not later than 1s after the Virtual Keyboard Trigger message. Note: an empty Server Cut Text message indicates that no initial server side text is available.
3. The VNC client MUST use the provided initial text within its virtual keyboard
4. The VNC client MUST send a VNC Client Cut Text message back to the VNC server on completion of the text entry at the client side.
5. The VNC client MUST NOT send any VNC key event prior to the VNC Client Cut Text message.
6. The VNC server MUST use the returned text entry as the confirmed text entry. Note: an empty Client Cut Text message indicates that the text entry has been canceled.
7. The VNC Server MUST send a VNC Virtual Keyboard Trigger message removing the virtual keyboard, not later than 1s after the reception of the VNC Client Cut Text message. Note: in case further text entry is needed, the VNC server MAY send a new VNC Server Cut Text message instead of the VNC Virtual keyboard Trigger message, removing the keyboard.

5.7 Device Status Messages

The Device Status and Device Status Request message pair provides the client with status information of specific device settings at the server and the ability to set them. The server MUST inform the client about any status change. The message flow is shown in Figure 14.

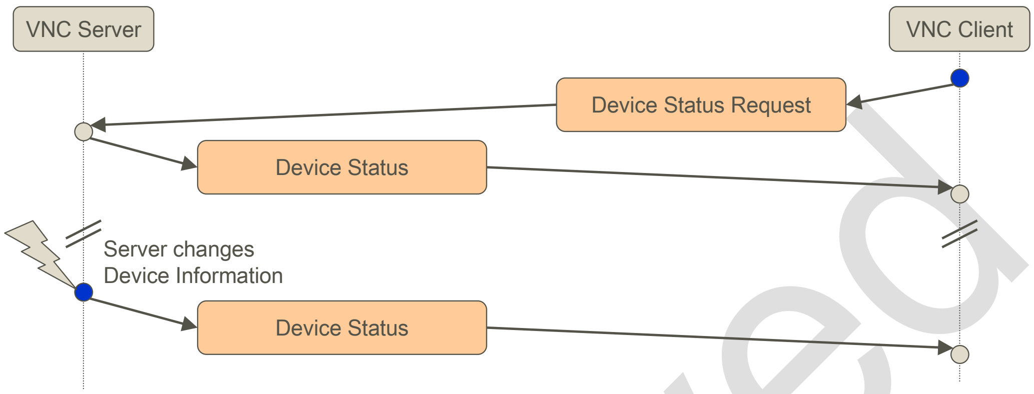

Figure 14: Example Device Status Message Flow

The VNC Server MUST immediately respond to a Device Status Request.

The Device Status message is given in Table 20.

# 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 (Do not allow key and pointer event entry at the CE device)
		[3:2]	Device-lock (In device-lock state, the MirrorLink Server is locked and MAY NOT respond to remote key and pointer events. Note: The User MAY need to enter a PIN code to un-lock the device; PIN code entry MAY NOT be possible via the MirrorLink Client)
		[5:4]	Screen saver (Do not show content on server display, e.g. dim-ming the display backlight)
		[7:6]	Night mode
		[9:8]	Voice control input on MirrorLink Server[10]

[10] The MirrorLink server MAY use these bits to indicate to the MirrorLink client, that it expects voice control input to be enabled or disabled.

# bytes	Type	Value	Description
		[11:10]	Microphone input on MirrorLink Client[11] (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 MUST be treated as voice command audio, if Voice Control Input status is enabled.
		[17:16]	Driver Distraction Avoidance (00 = unknown: MUST NOT use 01 = reserved: MUST 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) (000 = unknown, 001, 010, 011 = reserved, 100 = 0º, 101 = 90º, 110 = 180º, 111 = 270º)
		[28:27]	Framebuffer orientation (00 = unknown, 01 = reserved, 10 = Landscape, 11 = Portrait)

Table 20: Device Status Message

The Device Status Request message is given in Table 21.

# bytes	Type	Value	Description
1	U8	128	Message-type
1	U8	12	Extension-type
2	U16	4	Payload length
4	U32	*Bit*	*Status of Device Features (00 = ignore, 01 =* reserved *10 = disable, 11 = enable))*
		[1:0]	Key-lock (block key entry on the device)
		[3:2]	Device lock (bring device into device-lock state)
		[5:4]	Screen saver (power-down the device screen)
		[7:6]	Night mode (run application in night mode; device status flag provided via the Common API)
		[9:8]	Voice input (route the incoming audio stream to a voice recognition engine on the mobile device)[12]

[11] 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 MUST NOT be used to indicate end or start of phone call for BT HFP case.

[12] The MirrorLink client MUST 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 MUST use the BT HFP voice activation mechanism (AT + BVRA command as specified in [6]) instead.

# bytes	Type	Value	Description
		[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 (ena-bled), or non-restricted driving mode (disabled))
		[26:24]	Absolute Framebuffer rotation (clock-wise) (000 = ignore, 001, 010, 011 = reserved 100 = 0º, 101 = 90º, 110 = 180º, 111 = 270º)
		[28:27]	Framebuffer orientation (00 = ignore, 01 = reserved, 10 = Landscape, 11 = Portrait)

Table 21: Device Status Request Message

The VNC Server MAY ignore some or all device features as set in the Device Status Request message, if not specified otherwise below. The client SHOULD be able to detect this from the received Device Status message, following the original Device Status Request 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 MUST either report its current setting or report "`unknown`".

The VNC Client SHOULD NOT use the "`reserved`" value in any of the Device Status Request 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 Device Status message features. If the VNC Client receives a Device Status feature with the "`reserved`" value, it SHOULD consider the value being "`unknown`".

The MirrorLink Client SHOULD send device status request 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 MUST send a Device Status Request 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 [22]).

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

Framebuffer Orientation

If an application requires a change in the display orientation (e.g. from portrait to landscape mode), the VNC Server MUST send a Desktop Size Pseudo Encoding message, followed by a Device Status Message, announcing the new display orientation, if the MirrorLink Client has enabled the "`Server-side framebuffer orientation switch used`" flag within the Client Display Configuration message.

The VNC Client MAY send Framebuffer Blocking Notification with the reason "`UI layout not supported`". In that case, the VNC Server MUST change the layout back to the original orientation either within the current application or outside the current application (i.e. terminating the application and going back to a default home screen); the VNC Server MUST send a Desktop Size Pseudo Encoding message again, followed by a Device Status message, with the original framebuffer orientation.

If an application requests a PORTRAIT display orientation, the VNC Server MUST ignore the request, if the MirrorLink Client has disabled the `"Server-side framebuffer orientation switch used"` flag within the Client Display Configuration message.

The MirrorLink Server and Client MUST support Landscape orientation mode at any time. The MirrorLink Server and Client MAY support Portrait mode as well. The MirrorLink Server MUST start in Landscape orientation.

The MirrorLink Server MUST prevent automatic orientation change, due to accelerometer or other orientation sensor readings. The MirrorLink Server MAY allow the user to intentionally change the orientation during park mode. This is not considered an automatic orientation change.

The MirrorLink Server MUST report `"Landscape"` or `"Portrait"` values, even if the MirrorLink Client has set the flag's value to `"ignore"`.

Rotation

The MirrorLink Server MUST prevent automatic flipping (180 degree rotation), due to accelerometer or other sensor readings.

The MirrorLink Server MUST report `"0", "90", "180"` or `"270"` values, even if the MirrorLink Client has set the flag's value to `"ignore"`.

Voice Input & Microphone Input

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

The MirrorLink Client MUST respond with `"disabled"` if the Mic Input is currently occupied internally.

The MirrorLink Server and MirrorLink Client MUST 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 MUST use the `"ignore"` or `"unknown"` values.

Device Lock

The MirrorLink Server MUST send a Device Status message, with Device Lock enabled, prior going into the Device Lock.

5.8 Content Attestation Messages

The Content Attestation Request and Response 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 favorable 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 15.

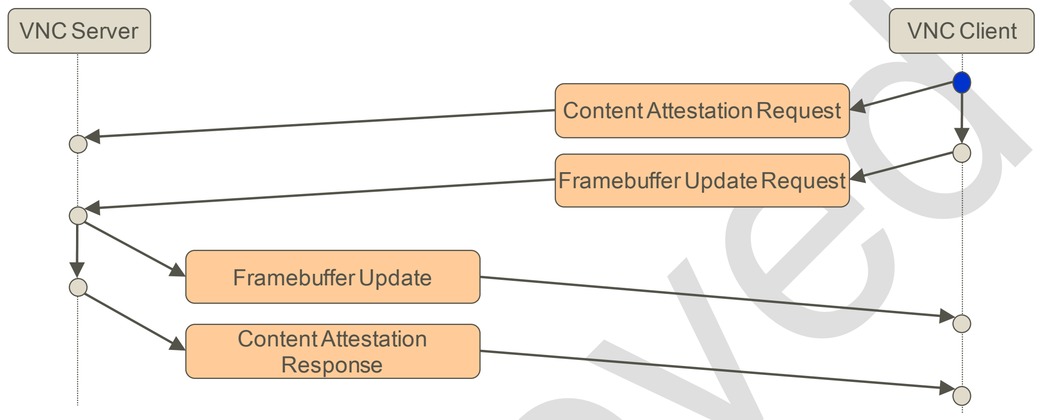
VNC Server
VNC Client
Content Attestation Request
Framebuffer Update Request
Framebuffer Update
Content Attestation Response

Figure 15: Example Content Attestation Message Flow

If the MirrorLink Server provides an `applicationPublicKey` bound to the VNC Server during a DAP session, the MirrorLink Server MUST support VNC Content attestation, at least providing a Content Attestation Response messages in response to a Con-tent Attestation Request message.

The VNC Server MUST immediately respond to a Content Attestation Request, after sending the next Framebuffer Update. The VNC Server MUST include full context information to the framebuffer update message, even if the context has not changed.

The VNC Client MUST send the Content Attestation Request message right before the next Framebuffer Update Request message. This is REQUIRED as the MirrorLink server MUST provide the Content Attestation Response together with the Framebuffer Update.

The Content Attestation Response message is given in Table 22.

# 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, that 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 client to decide whether such attestation is acceptable.
		[0]	SignedInfo includes context information pseudo encoding as provided within the last framebuffer update

# bytes	Type	Value	Description
			Note: Signature is calculated over all context information pseudo encoding rectangles (as defined in Table 31) in the Framebuffer Update 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 Framebuffer Update 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 content attestation response 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
18 - 86	Array of U8		SignedInfo (Size dependent of SignedInfo Flag)
32	Array of U8		(Optional) Signature

Table 22: Content Attestation Response Message

The Signature element in Table 22 is a signature over SignedInfo element described below in Table 23. The used signature algorithm is defined in Content Attestation Request message.

# bytes	Type	Value	Description
16	Array of U8		Nonce as provided by the MirrorLink client in Content Attestation Request (Table 24)
2	U16		SignedInfo flag that defines the attested content. The possible values are defined in Table 22
32	Array of U8		(Optional) SHA-256 hash of context information pseudo encoding, as provided within the last framebuffer update (as defined in Table 22) 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 22) Included if SignedInfo flag has bit 1 set.

4	U32	(Optional) Number of framebuffer pixels sent since previous content attestation response message (as defined in Table 22). 32-bit unsigned integer in network byte order. Included if SignedInfo flag has bit 2 set.

Table 23: SignedInfo Element for HMAC-SHA-256 Signature Type

The Content Attestation Request message is given in Table 24.

# 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 MUST be attested and included into the hash ('1' = include, '0' = do not include)*
		[0]	Include context information pseudo encoding, as defined in Table 22
		[1]	Include last framebuffer update, as defined in Table 22
		[2]	Include number of pixels sent since previous content attestation response message, as defined in Table 22
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 22.
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 Content Attestation Response messages, until a new key is provided in next Content Attestation Request.
N	Array of U8		(Optional) Session key. The client MUST 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.

Table 24: Content Attestation Request Message

5.9 Framebuffer Blocking Notification

The Framebuffer Blocking Notification message allows the MirrorLink client to notify the MirrorLink server, that certain framebuffer content delivered to it (as part of a Framebuffer Update message) has been blocked.

The original VNC protocol does not provide a mechanism to inform the MirrorLink server about the blocking. If the server was aware, it could take further actions. The Framebuffer Notification message will provide that feedback, as shown in Figure 16below.

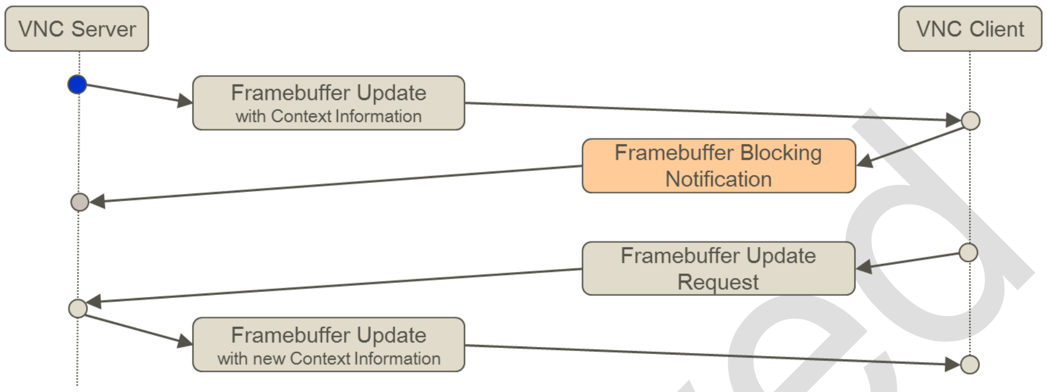

Figure 16: Example Framebuffer Blocking Notification Message Flow

The Framebuffer Blocking Notification message provides the MirrorLink Server with the rectangle area being blocked. This area and the application identifier MUST correspond with the rectangle information from the Context Information Pseudo Encoding. The MirrorLink client MUST send a Framebuffer Blocking Notification message for each rectangle, which is being blocked. I.e. if no rectangle is getting blocked, no Framebuffer Blocking Notification message is sent.

MirrorLink Client is blocking the MirrorLink Foreground Application

The MirrorLink Client MAY block the MirrorLink Server's current foreground application. Blocking in the context means that the application is unknown, is not providing the required certification status (in park or drive mode) or is not falling into the supported application or content categories (in park mode). The MirrorLink Client MUST NOT block any Drive-certified application in Drive Mode and MUST NOT block any Drive- or Base-certified application in Park Mode.

On reception of a Framebuffer Blocking Notification message, the MirrorLink server MUST respond with one of the following actions, in the given order:

1. Bring the available MirrorLink Server Home Screen application into the foreground. The certification status of the Home Screen application MUST support the current driving mode of the MirrorLink Client, when moving it into the foreground.

   MirrorLink Server MUST inform the user about the blocking.

2. Bring most recent running certified application into the foreground. The certification status of the application MUST support the current driving mode of the MirrorLink Client, when moving it into the foreground.

   MirrorLink Server MUST inform the user about the blocking.

3. Provide context information, containing at least an application category of `"Switch to MirrorLink Client native UI" (0xF000FFFF)`. Note: The MirrorLink Server MAY intentionally terminate the VNC session in case the MirrorLink Client does not switch to native UI and continues blocking the current framebuffer.

   MirrorLink Client MUST 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 SHOULD allow for some grace period of at least 1s, prior taking other actions.

The MirrorLink Server's decision tree, when the MirrorLink Client is blocking a foreground application is shown in Figure 17.

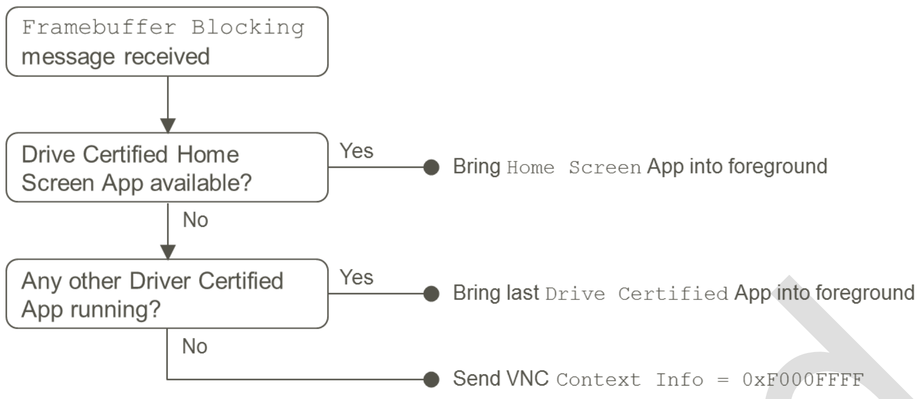
Framebuffer Blocking message received
Drive Certified Home Screen App available?
Yes
Bring Home Screen App into foreground
No
Any other Driver Certified App running?
Yes
Bring last Drive Certified App into foreground
No
Send VNC Context Info = 0xF000FFFF

Figure 17: Decision Tree, blocking a Foreground Application

The MirrorLink Client MUST continue sending `Framebuffer Update Request` messages.

MirrorLink Client is blocking a MirrorLink Overlay Frame

The MirrorLink Client MAY block an overlay over the MirrorLink Server's current foreground application. The overlay is originating from a background application or from the platform, i.e. the current foreground application is not changed. VNC context information of the overlay MUST be provided in the context of the originating application.

Blocking in the context means that the overlay is unknown, is not providing the required certification status (in park or drive mode) or is not falling into the supported application or content categories (in park mode). The MirrorLink Client MUST NOT block any overlay from Drive-certified application in Drive Mode and MUST NOT block any overlay from Drive- or Base-certified application in Park Mode.

On reception of a Framebuffer Blocking Notification message, the MirrorLink server MUST respond with one of the following actions, in the given order:

1. Move the overlay to the background the application creating the overlay. Note: that this assumes the overlay can be detected by the MirrorLink Server.

   MirrorLink Server MUST inform the user about the blocking.
2. Terminate the application, creating the overlay. Note: that this assumes the overlay can be detected by the MirrorLink Server and associated to an application by the MirrorLink Server.

   MirrorLink Server MUST inform the user about the blocking.
3. In case park mode is enabled, leave the overlay, despite the framebuffer blocking messages.

   User MUST NOT be informed about the blocking.
4. Provide context information, containing at least an application category of `"Switch to MirrorLink Client native UI" (0xF000FFFF)`. Note: The MirrorLink Server MAY intentionally terminate the VNC session in case the MirrorLink Client does not switch to native UI and continues blocking the current framebuffer.

   MirrorLink Client MUST inform the user about the blocking.

The MirrorLink Server's decision tree, when the MirrorLink Client is blocking an overlay of a background application is shown in Figure 18**Error! Reference source not found.**.

Switching to the MirrorLink Client screen, in the last step, may not resolve the problem, as the overlay MAY stay on the MirrorLink Server's screen, but no uncertified content is shown on the MirrorLink Client display at this point.

The MirrorLink Client MUST continue sending `Framebuffer Update Request` messages.

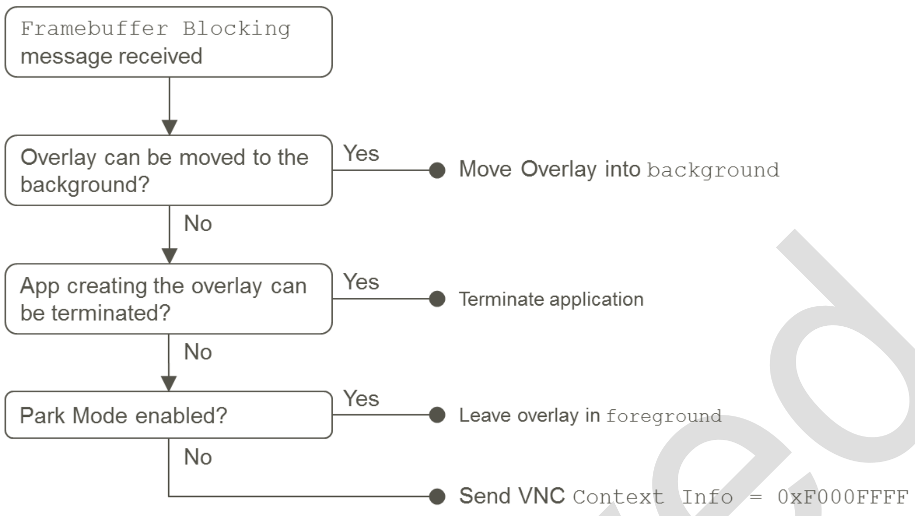
Framebuffer Blocking message received
Overlay can be moved to the background?
Yes
Move Overlay into background
No
App creating the overlay can be terminated?
Yes
Terminate application
No
Park Mode enabled?
Yes
Leave overlay in foreground
No
Send VNC Context Info = 0xF000FFFF

Figure 18: Decision Tree, blocking an Overlay of a Background Application

Implementation Notes:

- The MirrorLink Server MAY NOT be able to detect Framebuffer overlays from background applications or platform functions, as detailed in the platform specific specifications. In that case, the overlay will be reported as part of the current foreground application.
- The MirrorLink Server MAY NOT be able to move into the background, as detailed in the platform specific specifications. In that case, the overlay MAY stay in the foreground.

MirrorLink Server is switching the Framebuffer Orientation

The MirrorLink Client MUST NOT block a framebuffer for the reason "`UI layout not supported`" in case the MirrorLink Server is switching to Landscape mode.

The MirrorLink Client MUST block a framebuffer with the reason "`UI layout not supported`" in case the MirrorLink Server is switching to Portrait mode, while being in Drive mode.

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, e.g. as the user is launching an embedded application.

If the MirrorLink Client responds with a reason for blocking "`UI not visible on remote display`"[13], the MirrorLink Server MUST keep the current application in the foreground. The MirrorLink Server SHOULD consider the application being in the background on the MirrorLink Client's screen.

The VNC client MUST NOT send `Framebuffer Update Request` messages, as long as the MirrorLink Server's UI is not visible on the remote display. The application SHOULD be considered being in the foreground again once the MirrorLink Client's VNC client starts sending a `Framebuffer Update Request` message again.

The MirrorLink Client and Server MUST keep the VNC session alive[14].

[13] Some older MirrorLink Clients MAY also additionally use the deprecated reason bit 8. The MirrorLink Server SHOULD ignore this reason bit.

[14] Some older MirrorLink Clients or Server MAY terminate the VNC session, when the MirrorLink Server's Framebuffer is moved to the background. This MUST NOT be considered a misbehaving device.

User Information about Framebuffer Blocking

The User MUST 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.

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 Framebuffer Blocking Notification message is given below.

# 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 (MUST be identical to the unique application ID provided within the Context Information Pseudo Encoding message)
2	U16	*Bit*	*Reason for blocking (‘1’ = reason, ‘0’ no reason)*
		[0]	Not allowed content category • Use of this reason flag is **deprecated**.
		[1]	Not allowed application category The MirrorLink Client MUST set this reason bit for the following reason: • MirrorLink Client does not support MirrorLink aware applications with given application category in park mode.
		[2]	Not sufficient content trust level • Use of this reason flag is **deprecated**.
		[3]	Not sufficient application trust level The MirrorLink Client MUST set this reason bit for the following reason: • Application is blocked for certification status reason (e.g. a non-drive application in drive mode).
		[4]	Content rules not followed • Use of this reason flag is **deprecated**.
		[5]	Not allowed application ID The MirrorLink Client MUST set this reason bit for one of the following reasons: • AppID of the blocked application is set to “`0x00`” • AppID of the blocked application has not been included in the UPnP application listing. • Application is blocked for certification status reason (e.g. a non-drive application in drive mode).
		[8]	UI not in focus on remote display • Use of this reason flag is **deprecated**.
		[9]	UI not visible on remote display The MirrorLink Client MUST set this reason bit for the following reason:

# bytes	Type	Value	Description
			• MirrorLink Client has moved the MirrorLink Server's framebuffer into the background.
		[10]	UI layout not supported (after a Desktop Size Pseudo Encoding)

Table 25: Framebuffer Blocking Notification Message

The Framebuffer Blocking Notification message MUST correspond to the last received Framebuffer Update message with Context Information Pseudo Encoding. The VNC Client MUST send the Framebuffer Blocking Notification message before any subsequent Framebuffer Update Request message.

The VNC Server MAY receive the VNC Framebuffer Blocking Notification message only after the VNC Client has already made the "`UI not visible on remote screen`". The VNC Client MUST send the VNC Blocking Notification message immediately after receiving the next Framebuffer Update message following the removal of the UI from its screen.

In case the MirrorLink Client is switching to Drive Mode, which causes the blocking of the current foreground application, the MirrorLink Client MUST do one of the following:

1. Stay within the MirrorLink experience, but terminate the current foreground application, or
2. Stay within the MirrorLink experience, but launch a drive-certified application, or
3. Leave the MirrorLink experience and switch back to the MirrorLink Client's native screen, or
4. Stay within the MirrorLink experience, but immediately send a VNC Blocking Notification message; i.e. there is no need to wait until the reception of the next `Framebuffer Update` message.

In case of option 1, 2 or 3, the MirrorLink Client MUST inform the user about the action, as specified in the user notification requirement above.

The Framebuffer Blocking Notification sequence for option 4, which allows the MirrorLink Server to respond to the blocking, is shown in Figure 19.

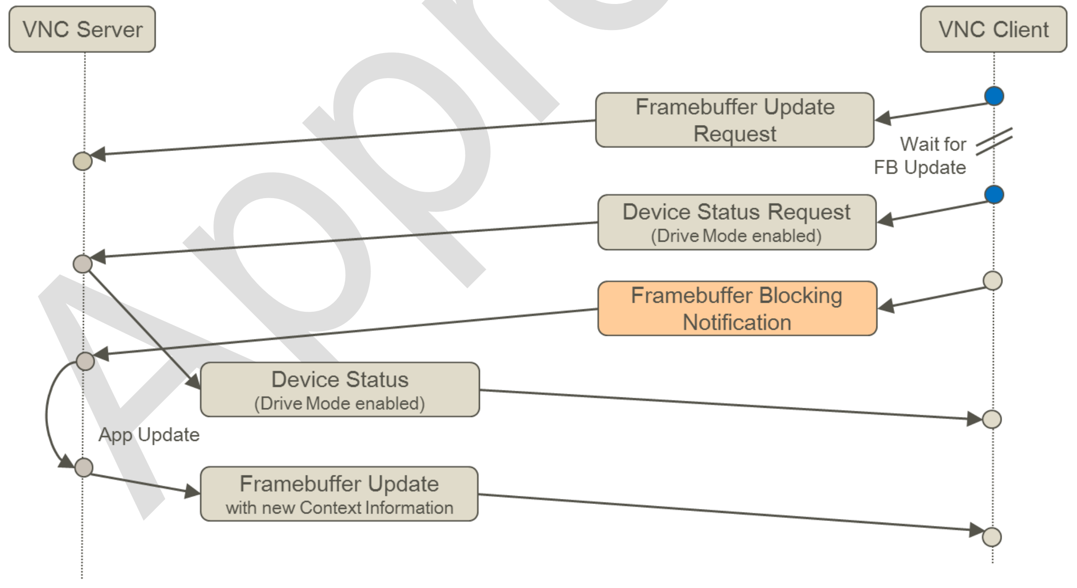
VNC Server
VNC Client
Framebuffer Update Request
Wait for FB Update
Device Status Request (Drive Mode enabled)
Framebuffer Blocking Notification
Device Status (Drive Mode enabled)
App Update
Framebuffer Update with new Context Information

Figure 19: Framebuffer Blocking Notification Message Flow due to enabling Drive Mode

Exception:

Some existing MirrorLink 1.1 Servers MAY change the application state, when receiving a Framebuffer Blocking Notification for the reasons "`UI not visible on remote screen`" or "`UI not in focus on remote screen`". A MirrorLink Client MAY therefore refrain from sending a Framebuffer Blocking Notification message, when the MirrorLink Server's framebuffer is not visible or not in focus. It MUST refrain from requesting new Framebuffer Update Request message though. This exception is given only to MirrorLink Clients, connected to a MirrorLink 1.0 Server.

Handling of Overlays

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

Handling of Applications Seeking Foreground Status

The MirrorLink Server MUST 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 [26], 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 [26].
- Presidential Alerts during a national emergency.

5.10 Audio Blocking Notification

The Audio Blocking Notification 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 server was aware it could take further actions. The Audio Blocking Notification message will provide that feedback, as shown in Figure 20 below.

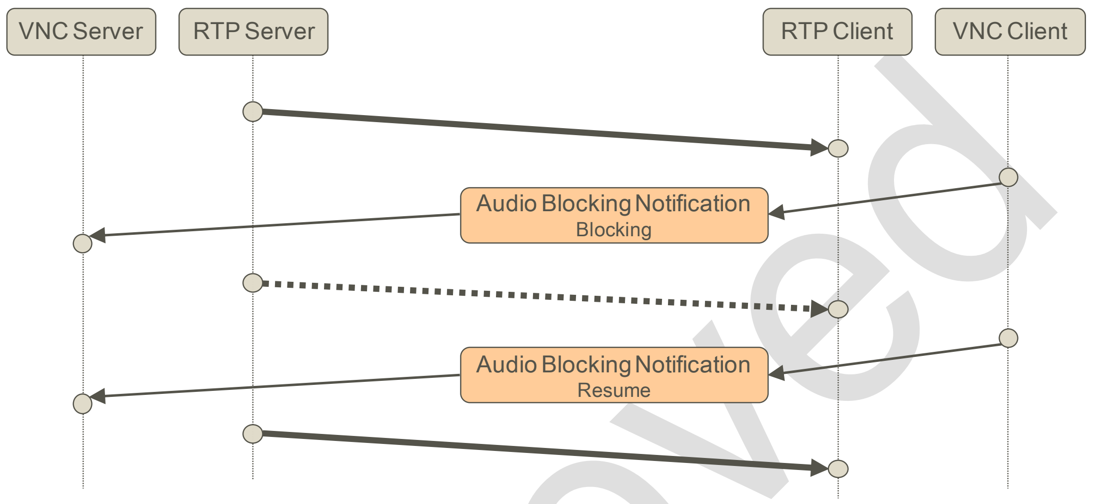

Figure 20: Example Audio Blocking Notification Message Flow

The Audio Blocking Notification message provides the MirrorLink server with the application identifier whose stream will be blocked. The application identifier MUST correspond to the advertised identifier in the UPnP application listing. The same application identifier MUST be used within the RTP extension header, specified in [20]. The MirrorLink client MUST send an Audio Blocking Notification 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 Audio Blocking 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 MUST send an Audio Blocking messages for each RTP audio stream it wants being removed.

The MirrorLink Client MUST NOT block audio streams from drive-certified application in Drive Mode and MUST NOT block audio streams from Drive- or Base-certified application in Park Mode, unless higher-priority native MirrorLink Client audio is playing or the user intentionally muted the audio from a certified application. The MirrorLink Client SHOULD block Audio Sources from un-certified applications, while in Drive Mode.

On reception of an audio blocking notification message, the MirrorLink Server MUST respond with the following actions:

- The MirrorLink Server MUST notify CCC or member certified applications, and SHOULD notify other applications via the Common API about the blocking.

- The MirrorLink Server MUST stop sending RTP packets containing foreground[15] content from blocked applications[16].
- The MirrorLink Server MUST NOT block audio from CCC or member certified applications, unless they have been explicitly blocked.

Implementation Notes:

- The MirrorLink Server MAY NOT be able to remove audio from non-certified applications, as detailed in the platform specific specifications. In that case, the audio will continue playing.

The MirrorLink Server's decision tree, when the MirrorLink Client is blocking audio is shown in Figure 21.

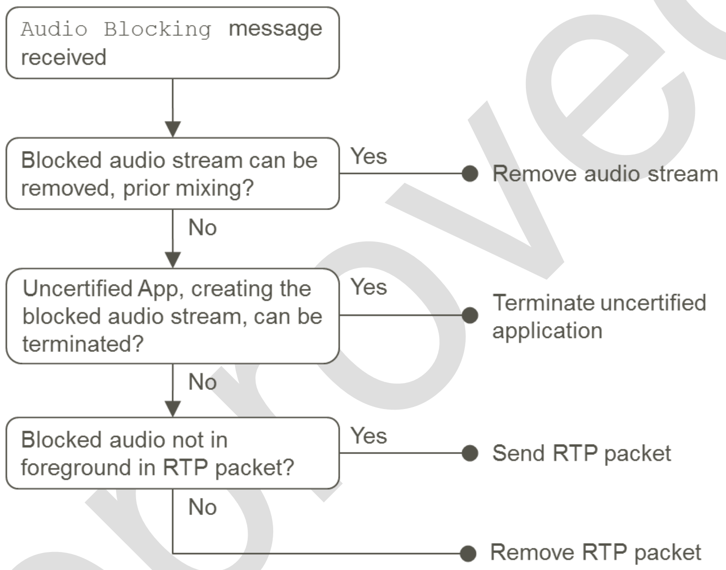
Audio Blocking message received
Blocked audio stream can be removed, prior mixing?
Yes
Remove audio stream
No
Uncertified App, creating the blocked audio stream, can be terminated?
Yes
Terminate uncertified application
No
Blocked audio not in foreground in RTP packet?
Yes
Send RTP packet
No
Remove RTP packet

Figure 21: Decision Tree, blocking Audio

The MirrorLink Server and Client MUST keep the RTP session and any VNC session alive,[17]

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 MUST send Audio Blocking messages for each Audio Source, reported via the RTP extension header. The MirrorLink Client MAY send Audio Blocking messages to additional applications from the UPnP Application listing, which are currently not providing audio.

On reception of an audio blocking notification message with global mute, the MirrorLink Server MUST respond with the following actions:

[15] 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.

[16] 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.

[17] 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 MUST NOT be considered a misbehaving device.

- The MirrorLink Server MUST notify CCC or member certified applications, and SHOULD notify other applications via the Common API about the blocking.
- The MirrorLink Server MUST 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 MUST keep the RTP session and any VNC session alive,[18]

MirrorLink Client is unblocking the MirrorLink selected or all Audio Streams

The MirrorLink Client MUST send an Audio Blocking Notification 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 Audio Blocking 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 MAY NOT send an audio unblocking message. In this case, the user will need to manually resume the audio playback.

On reception of an audio blocking notification message, the MirrorLink Server MUST respond with the following actions:

- The MirrorLink Server MUST notify CCC or member certified applications, and SHOULD notify other applications via the Common API about the unblocking.
- The MirrorLink Server MUST resume sending RTP packets containing only content from unblocked applications, once audio becomes available again.

The Audio Blocking Notification message is given below in Table 26.

# 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 The MirrorLink Client MUST set this reason bit for one of the following reason: • MirrorLink Client does not support Audio from MirrorLink aware applications with given application category.
		[1]	Not sufficient application trust level The MirrorLink Client MUST set this reason bit for one of the following reason: • Audio is blocked for certification status reason (e.g. a non-drive application in drive mode).
		[2]	Not allowed application ID

[18] 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 MUST NOT be considered a misbehaving device.

# bytes	Type	Value	Description
			The MirrorLink Client MUST set this reason bit for one of the following reasons: • AppID of the blocked application is set to "`0x00`" • AppID of the blocked application has not been included in the UPnP application listing. • Application is blocked for certification status reason (e.g. a non-drive application in drive mode).
		[3]	Global audio muted The MirrorLink Client MUST 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 MUST set this reason bit for one of the following reason: • User has muted the audio stream.

Table 26: Audio Blocking Notification Message

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 Audio Blocking Notification is received, which indicates the blocking of an audio stream, otherwise the application is removed from the blacklist. On removal or addition, the affected applications is always informed via a specific Common API callback. This is shown in Figure 22 below.

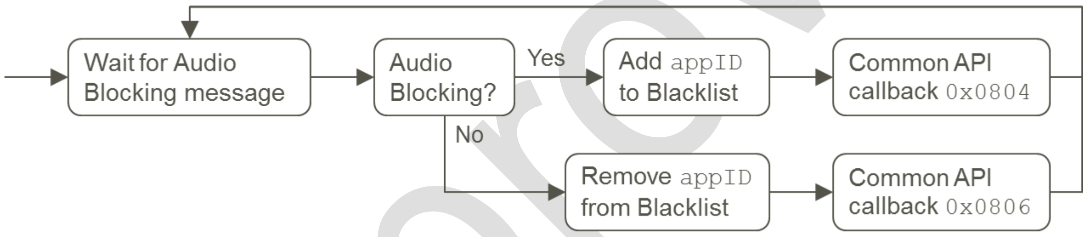
Wait for Audio Blocking message
Audio Blocking?
Yes
Add appID to Blacklist
Common API callback 0x0804
No
Remove appID from Blacklist
Common API callback 0x0806

Figure 22: Analyzing 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 analyzed, 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 behavior is shown in Figure 23 below. 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.

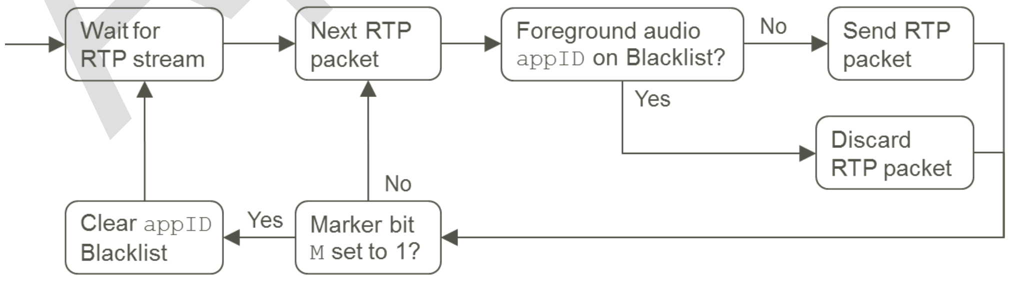

Figure 23: Analyzing the Audio Packets

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

5.11 Touch Event

The support for the Touch Event message is optional. The MirrorLink client SHOULD NOT send Touch Event messages, if it has disabled Bit [1] of the *Pointer related* configuration vector in the Client Event Configuration message. The MirrorLink server MUST ignore Touch Event messages, if the server or client has disabled Bit [1] of the *Pointer related* configuration vector in the Server or Client Event Configuration message.

The Touch Event 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 this specification is shown in Figure 24 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.

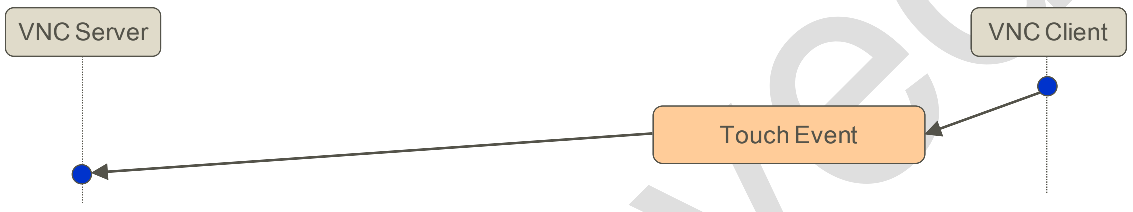

Figure 24: Touch Event Message Flow

The Touch Event message is given in Table 27.

# bytes	Type	Value	Description
1	U8	128	Message-type
1	U8	20	Extension-type
2	U16	1 + N*6	Payload length
1	U8	N	Number of individual events
N*6	Array of 6 bytes		Description of individual event

Table 27: Touch Event Message

Each individual event is described in Table 28 the following way:

# 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.

Table 28: Description of Individual Events

The VNC Client MUST 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 Client and Server Event Configuration message pair. Each event MUST be completed, i.e. each press event MUST be later followed by a release event.

The VNC Server MUST ignore a touch release event, if no touch-press event has been received before for the same event identifier. The VNC Server MUST 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 Client Event Configuration message, indicates the maximum pressure. The VNC Client SHOULD NOT provide pressure levels outside the client's pressure mask. The VNC Server MUST 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 server.

The following Table 29 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 Client Event Configuration messages. Note: It is implementation specific, whether the MirrorLink Server will need to adapt the received 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
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

Table 29: Adaptation of Touch Event Pressure Values

Touch and pointer events SHOULD not be mixed.

5.12 Framebuffer Alternative Text

The support of this message is optional.

The Framebuffer Alternative Text message allows the MirrorLink server to provide textual information about the current application to the MirrorLink client. It is not the subject of this specification to define how and where the MirrorLink client will show the framebuffer alternative text. The alternative text MUST NOT overlay valid framebuffer content. The flow is shown in Figure 25.

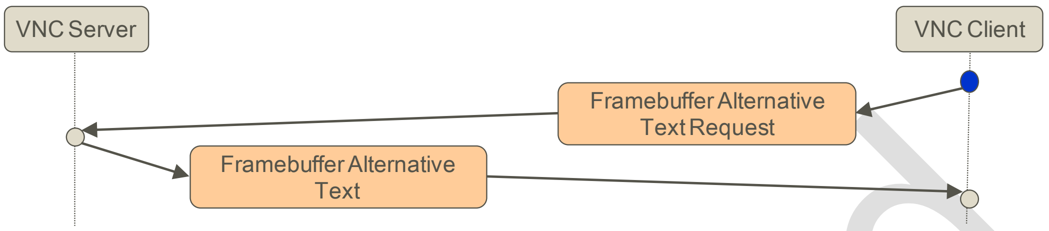
VNC Server
VNC Client
Framebuffer Alternative Text Request
Framebuffer Alternative Text

Figure 25: Example Framebuffer Alternative Text Message Flow

The Framebuffer Alternative Text message is given below in Table 30.

# bytes	Type	Value	Description
1	U8	128	Message-type
1	U8	21	Extension-type
2	U16	6 + N	Payload length
4	U32		Unique application id. Applications being advertised via UPnP, MUST match the advertised appID.
2	U16	N	Length of Textual Information. A zero value invalidates any previous meta information for that application.
N	Array of U8		Textual information (free text format)

Table 30: Framebuffer Alternative Text Message

The provided textual-information is valid for the identified application until it is either overridden from a new message or invalidated (i.e. the length of the textual information is zero). Multiple valid textual-information entries can exist in parallel for different uniquely identifiable applications.

The MirrorLink server MUST only send Framebuffer Alternative Text messages, if the MirrorLink client has enabled this feature, using the Framebuffer Alternative Text Request message, as given in Table 31:

# bytes	Type	Value	Description
1	U8	128	Message-type
1	U8	22	Extension-type
2	U16	2	Payload length
2	U16		Maximum length of the meta information. A zero length disables the feature.

Table 31: Framebuffer Alternative Text Request Message

The MirrorLink server MUST NOT send Framebuffer Alternative Text messages, with information longer than allowed from the MirrorLink client.

6 ADDITIONAL ENCODINGS AND PSEUDO ENCODINGS

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 32.

Type	Value	Description	Functionality	Support
Pseudo Encoding	-523	MirrorLink Encoding	Advertise the support of MirrorLink extension messages. Not used within Framebuffer Update messages.	Mandatory
Pseudo Encoding	-524	Context Information	Indicate context information within a Framebuffer Update 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	Framebuffer encoding, which includes a pixel format and a down-scale factor used within the framebuffer update	Optional

Table 32: Additional VNC Encodings

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

6.1 MirrorLink Pseudo Encoding

The MirrorLink Pseudo Encoding is used within the Set Encodings message to inform the server about the client support of the MirrorLink extension messages. The client MUST use MirrorLink Pseudo Encoding only within the Set Encoding message to indicate support for the MirrorLink extension messages. This Pseudo Encoding MUST NOT be used within any Framebuffer Update message.

The support for MirrorLink Pseudo Encoding is mandatory.

6.2 Context Information Pseudo Encoding

The Context Information Pseudo Encoding 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 level of trust in its correctness. If context information is available from different trust levels, the server MUST 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 MAY 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 26.

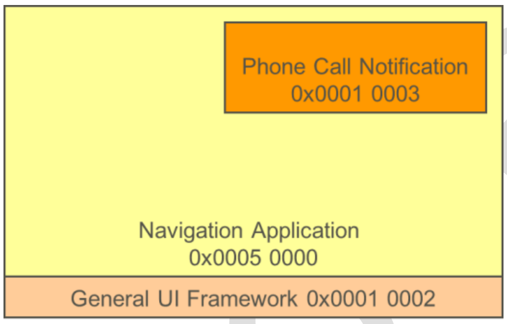

Figure 26: Context Information (Example)

Context information can be given either for the entire display at once, or for multiple, individual rectangular areas. Context information MUST be valid, until the next context information pseudo encoding. The server MUST provide context information for the entire display, i.e. the aggregation of the individual, related[19] rectangular areas MUST always cover the entire framebuffer, and never a partial framebuffer area alone. 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 MUST 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 MAY NOT be available, but the MirrorLink server MAY be able to provide more information about the framebuffer content type or vice versa.

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

The whole pseudo encoding rectangle is given in Table 33.

# 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

[19] Drive certification of applications for MirrorLink currently REQUIRES single full-screen application within a MirrorLink session. Therefore multiple rectangles MUST 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 MUST handle un-related context information rectangle.

# bytes	Type	Value	Description
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 MUST match the advertised appID. This field MAY be left empty (i.e. zero value).
2	U16		Trust Level for Application Category (see [22], Table 6-1)
2	U16		Trust Level for Content Category (see [22], Table 6-1)
4	U32		Application Category (see [22], Table 6-2)
4	U32		Content Category (see [22], Table 6-3)
4	U32	*Bit*	*Content rules, which are followed to prevent Driver Distraction. RuleIds are defined in* [22], Table 6-5.
		0	‘1’. Rule with ruleId 0 supported. ‘0’ otherwise
		1	‘1’: Rule with ruleId 1 supported. ‘0’ otherwise
		…	…
		31	‘1’: Rule with ruleId 31 supported. ‘0’ otherwise

Table 33: Context Information Pseudo Encoding

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 **[23]**.

The VNC Server MUST provide context information within the first RFB Framebuffer Update message. Additionally, the VNC Server MUST 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 Context Information Pseudo Encoding for an advertised application differs from the context information, provided within the UPnP application listing, the VNC Client MUST use the information from the Context Information Pseudo Encoding.

If the VNC client receives a Context Information Pseudo Encoding message, with an application category set to “Switch to MirrorLink Client native UI” (0xF000FFFF) the MirrorLink Client MUST switch to a native user-interface or intentionally disconnect the VNC session.

6.3 Desktop Size Pseudo Encoding

The Desktop Size Pseudo Encoding rectangle within a Framebuffer Update message indicates to the VNC Client, that the VNC Server has changed its framebuffer resolution. The format of Desktop Size Pseudo Encoding is specified in the RFB specification [1]. A Framebuffer Update message, containing a Desktop Size Pseudo Encoding message MAY include Content Information Rectangle(s) and Framebuffer Data rectangle(s).

The VNC Server SHOULD skip framebuffer data rectangles within a Desktop Size Pseudo Encoding message, even if the VNC Client originally requested a non-incremental Framebuffer Update.

The VNC Server SHOULD skip context information rectangles within a Desktop Size Pseudo Encoding message, even if the VNC Client originally requested a non-incremental Framebuffer Update.

If the VNC Server includes framebuffer data rectangles within a Desktop Size Pseudo Encoding message, the data rectangles MUST NOT be clipped to a resolution smaller than requested, in case of a non-incremental Framebuffer Update.

The MirrorLink Client MUST support Desktop Size Pseudo Encoding.

The MirrorLink Client MAY request a new framebuffer update, before the last framebuffer update message has been fully decoded. Therefore the VNC Client MAY miss a Desktop Size Pseudo Encoding rectangle, and MAY therefore request a Framebuffer Update for the “old” framebuffer dimension.

Therefore, the MirrorLink Server SHOULD send a Framebuffer Update message with a Desktop Size Pseudo Encoding message, if the MirrorLink Client has requested a Framebuffer Update with a framebuffer area exceeding the MirrorLink Server’s framebuffer resolutions.

In any case, the MirrorLink Client MUST ignore the Context Information and Framebuffer Data rectangles in response to a Framebuffer Update Request message for the old Server framebuffer resolution, i.e. after the MirrorLink Server has changed its framebuffer resolution.

6.4 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 MUST 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 color value. A run MUST NOT span across multiple lines. If the length of a run is bigger than maximum allowed run-length, the run MUST be split into smaller runs. The minimum run length is 1. The whole Run-Length Encoding is given in Table 34.

Compared to 16-bit color depth, a 12-bit or 10-bit color depth can decrease the amount of transferred data as there is no additional byte used for run length. In addition, the reduced color depth helps to increase run length. It is RECOMMENDED to use 12-bit and 10-bit color depth together with run-length encoding, in case of limited available bandwidth, e.g. in case of WLAN.

The basic structure is defined in the RFB 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 35.

Table 34: Run-Length Encoding within a Framebuffer Update Message

The run-length encoding MUST be done line-by-line. The encoding for each individual line is given in the Table 35. Runs MUST NOT span across multiple lines.

# 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 Color depth

Table 35: Run-Length Encoding for individual Line

The number of bytes per run (K) is dependent on the color depth of the selected pixel format. The run-length encoding only considers the color value of the pixel. If the bits-per-pixel value is higher than the color depth, those additional bits are ignored. The encoding of an individual run is shown in Table 36.

# bytes	Type	Value	Description
K	Run-length Encoding[20]	*Bit*	*Run-length Encoding*
		[C+R-1:C]	Run length minus 1
		[C-1:0]	Color Value[21]

Table 36: Run-Length Encoding for Individual Run

[20] Endianess of the K-byte block follows the pixel's color value endianess, as set from the VNC Server (RFB Server Init message) or requested from the VNC Client (Set Pixel Format message)

[21] Color-value contains bits of the pixel value; E.g. in case of ARGB888, these are 24 bit containing the R, G and B values, in case of RGB 555, they are 15 bit; transparency bits are not included.

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

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

$$C_m = C \bmod 8$$

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

The maximum run-length is therefore at least 16 (4 bits). Table 37gives run-length encoding values for example color formats.

Color Format	Bits per Pixel	Color Depth	# bytes K	# bits for color value C	# bits for run length R
ARGB 888	32	24	4 (U32)	24	8
RGB 565	16	16	3 (U24)	16	8
RGB 555	16	15	3 (U24)	15	9
RGB 444	16	12	2 (U16)	12	4
RGB 343	16	10	2 (U16)	10	6
16-bit grayscale	16	16	3 (U24)	16	8
8-bit grayscale	8	8	2 (U16)	8	8

Table 37: Example Run-Length Encoding Values

6.5 Transform Encoding

The Transform Encoding allows the VNC Server to declare to the VNC Client that the server transforms the framebuffer's pixel data in particular ways prior sending it the VNC Client.

- The VNC Server MAY scale-down the framebuffer update
- The VNC Server MAY change the pixel format of pixel data

Transforming pixel data adaptively provides a large degree of flexibility in how to trade off parameters such as framebuffer data resolution and network bandwidth.

Unless the VNC Client includes Transform Encoding and MirrorLink Encoding in the Set Encodings message, the VNC Server MUST NOT transform framebuffer data and use Transform encoding. Transform encoded framebuffer updates MUST be included in regular Framebuffer Update messages.

The basic structure is defined in the RFB 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	-526	Transform Encoding type
N	Array of U8		Transform encoded data as given in the next table.

Table 38: Transform Encoding within a Framebuffer Update Message

The VNC Server can reduce framebuffer data and/or change the pixel format of the data, which results in reducing the data size. The transformation applied to the framebuffer data MUST be specified in the Transform Encoding section within the Framebuffer Update message, as defined in Table 39.

# bytes	Type	Value	Description
1	U8		Offset x-position $\Delta x_{resized}$
1	U8		Offset y-position $\Delta y_{resized}$
1	U8	*Value*	*Resize Factor (Value is identical to the bit position within Client Display Configuration message)*
		0	Resizing factor is 1
		1	Resizing factor is 1/2
		2	Resizing factor is 1/3
		3	Resizing factor is 1/4
		4	Resizing factor is 1/5
		5	Resizing factor is 1/6
		6	Resizing factor is 1/8
		7	Resizing factor is 1/10
		8	Resizing factor is 1/16
		9	Resizing factor is 1/32
		10	Resizing factor is 2/3
		11	Resizing factor is 3/4
1	U8	*Value*	*Pixel Format Value*
		0	32-bit ARGB 888

# bytes	Type	Value	Description
		8	24-bit RGB 888
		16	16-bit RGB 565
		17	16-bit RGB 555
		18	16-bit RGB 444
		19	16-bit RGB 343
		24	16-bit single color
		25	8-bit single color
4	S32		Encoding-type in transformed framebuffer
N	Array of U8		Framebuffer data in specified encoding type.

Table 39: Transform Encoding Structure

The VNC Server will reduce the width and height of the original framebuffer update area by a given resizing factor.

The VNC Server MAY apply low-pass filtering prior sending the pixels. The VNC Server MUST NOT use any resize factor, not supported from the VNC Client as indicated in the Client Display Configuration message. The VNC Client MUST support a resize factor of 1.

The offset ($\Delta x_{transform}$, $\Delta y_{transform}$) of the transform encoded framebuffer, defines the offset of the first transform encoded pixel, relative to the server's native framebuffer origin (upper left corner). The offset is based on the offset ($\Delta x_{original}$, $\Delta y_{original}$) of the original (non-transformed) framebuffer update plus an additional offset ($\Delta x_{resized}$, $\Delta y_{resized}$) coming from the resizing of the framebuffer prior to its transmission. The offset ($\Delta x_{resized}$, $\Delta y_{resized}$) defines the offset of the first transform encoded pixel, relative to the original framebuffer update's origin (upper left corner):

$$\Delta x_{transform} = \Delta x_{original} + \Delta x_{resized}$$

$$\Delta y_{transform} = \Delta y_{original} + \Delta y_{resized}$$

Both offsets MUST follow the rules that $\Delta x_{resized} < 1/ResizeFactor$ and $\Delta y_{resized} < 1/ResizeFactor$. The reduced width and height of the transformed framebuffer are given below:

$$Width_{transform} = \left\lceil \left(Width_{original} - \Delta x_{resized}\right) \times ResizingFactor \right\rceil$$

$$Height_{transform} = \left\lceil \left(Height_{original} - \Delta y_{resized}\right) \times ResizingFactor \right\rceil$$

$$\text{With } \lceil x \rceil = ceil(x)$$

The following figures shows an example, where the brown pixels are supposed to be transferred (original framebuffer update). As a result of the transform encoding only the orange ones are actually being transferred.

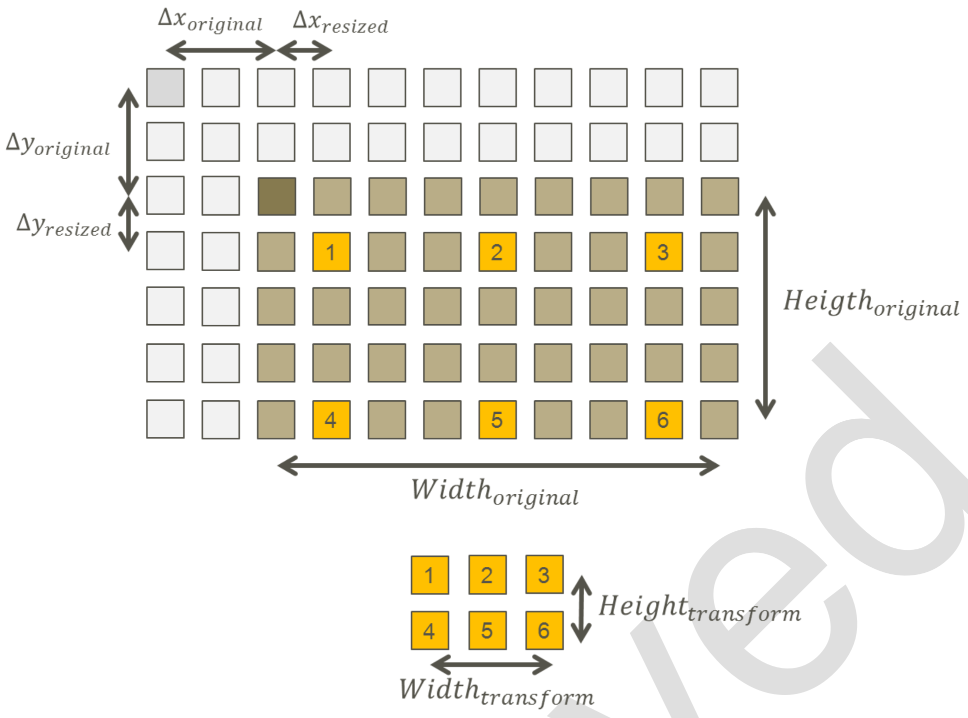

Figure 27: Example Transform Encoded Framebuffer Update

The VNC Server is able to change pixel format for the pixel data, as indicated in the Pixel Format entry of the Transform Encoding. The VNC Server MUST select a pixel format, supported from the VNC Client, as indicated in the VNC Client Display Configuration message.

If Transform encoded rectangles are mixed with other transform encoded rectangles with different resize levels or with regular non-transform encoded rectangles within the same Framebuffer Update message, Transform encoded ones MUST come prior to any regular non-transform encoded rectangles. Transform encoded rectangles MUST be ordered according to resize levels (higher resize levels first). This ensures that higher-quality framebuffer content, is not overlaid by lower-quality one.

Table 40 shows the resulting transform encoded framebuffer update (from the example, given in Figure 27).

# bytes	Type	Value	Description
2	U16	2	X-position of rectangle (top left corner)
2	U16	2	Y-position of rectangle (top left corner)
2	U16	9	Width of rectangle
2	U16	5	Height of rectangle
4	S32	-526	Transform Encoding type
1	U8	1	Offset x-position (to original x-position)
1	U8	1	Offset y-position (to original y-position)
1	U8	2	Resize Level Value
1	U8	16 (RGB565)	Pixel format Value
4	S32	0 (Raw)	Encoding-type in transformed framebuffer
12	Array of 6 RGB565 values	Framebuffer data in specified encoding type.

Table 40: Example Transform Encoded Framebuffer Update

7 REFERENCES

[1] Tristan Richardson, “The RFB Protocol”, RealVNC Ltd, Version 3.8, August 28, 2008.

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

[3] IETF, RFC 3551, “RTP Profile for Audio and Video Conferences with Minimal Control”, July 2003, http://www.ietf.org/rfc/rfc3551.txt

[4] USB CDC/NCM, “Universal Serial Bus Communications Devices Class Subclass Specifications for Network Control Model Devices”, Revision 1.0, April 30, 2009.

[5] BT SIG, “Simple Pairing”, White Paper, Core Specification Working Group, Revision V10r00, August 3, 2006.

[6] BT Specification, “Hands-free Profile 1.5”, Car Working Group, Revision V10r00, November 25, 2005.

[7] BT Specification, “Handset Profile”, Car Working Group, Revision V12r00, December 18, 2008.

[8] UPnP Specification, “UPnP™ Device Architecture”, Version 1.1, Revision October 15, 2008

[9] Bluetooth Assigned Numbers, http://www.bluetooth.org/assigned-numbers.htm

[10] Bluetooth Specification “Audio/Video Distribution Transport Protocol (AVDTP)”, Revision 1.00

[11] XML Signature Syntax and Processing, http://www.w3.org/TR/xmldsig-core/, W3C Recommendation 10 June 2008

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

[13] IETF, RFC 4648, The Base 16, Base 32 and Base 64 Data Encodings, October 2006. http://tools.ietf.org/html/rfc4648

[14] IETF, RFC 5280, Internet X.509 Public Key Infrastructure Certificate, May 2008. http://tools.ietf.org/html/rfc5280

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

[16] Trusted Computing Group. http://www.trustedcomputinggroup.org/

[17] Trusted Platform Module (TPM) specifications. http://www.trustedcomputinggroup.org/resources/tpm_main_specification

[18] Mobile Trusted Module (MTM) specifications. http://www.trustedcomputinggroup.org/resources/mobile_phone_work_group_mobile_trusted_module_specification_version_10

[19] ITU-T ASN.1 Encoding Rules. http://www.itu.int/ITU-T/studygroups/com17/languages/X.690-0207.pdf

[20] Car Connectivity Consortium, “MirrorLink – Audio Architecture”, Version 1.1; CCC-TS-012

[21] Car Connectivity Consortium, “MirrorLink – UPnP Server Device”, Version 1.1; CCC-TS-030

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

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

[24] Trusted Computing Group, “Credentials Profiles Specification 1.1”, May 2007. http://www.trustedcomputinggroup.org/resources/infrastructure_work_group_tcg_credential_profiles_specification

[25] Markus Kuhn, "Revision of the keysym specification", X.org Foundation, August 16, 2004, http://www.x.org/wiki/KeySyms

[26] Federal Emergency Management Agency (FEMA), Wireless Emergency Alerts (link accessed March 2015) https://www.fema.gov/wireless-emergency-alerts

APPENDIX A – KNOB CONFIGURATION

The shift and rotate operations of a 2D knob are given according to the coordinate system, given in Figure 28. The gray 2D area references the surface the knob is mounted to.

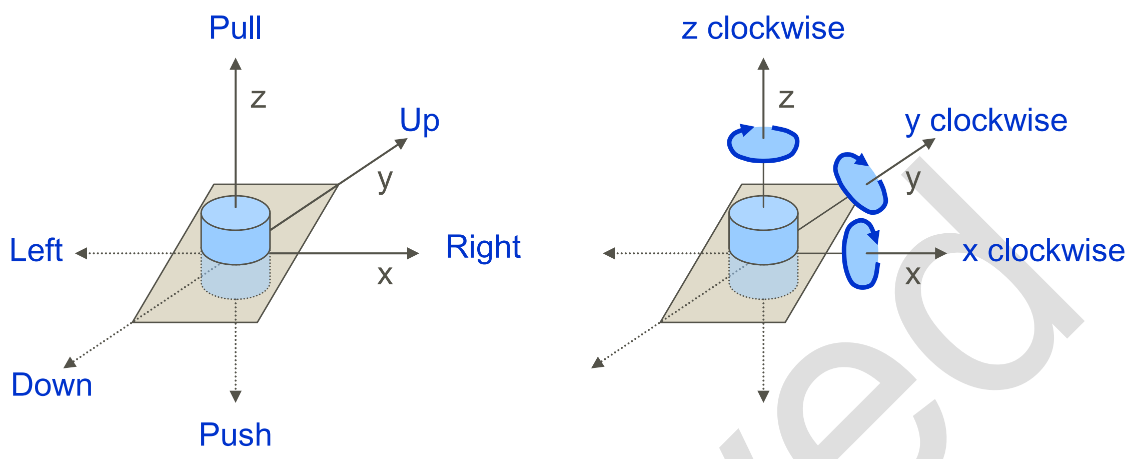

Figure 28: Coordinate System for Knob Events

The knob shift & rotate configuration is shown in Table 41, with allowed values for n in [0:3].

# 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

Table 41: Knob Shift and Rotate Configuration Settings

The 32-bit configuration vector can hold up to 4 knob configurations. The MirrorLink client and server MUST 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).

APPENDIX B – KEY EVENT MAPPING

The Key event mapping for different 2D knobs is shown in Table 42. The key event mapping for a particular head-unit knob n MUST be according the following format:

```
0 x 3 0 0 0   0 0 n m
```

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].

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	

Table 42: Head-Unit Knob Key Event Mapping

The Key event values for the ITU, Device, Function and Multimedia key events are given in Table 43.

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

Category	Mnemonic	KeySymValue	Description
	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 (applica-tion) menu
	Reserved	0x3000 0210	Reserved
		…	
		0x3000 02FF	
Function Keys	Function_Key_0	0x3000 0300	Soft and hard keys availa-ble on the client and server user interface
	Function_Key_1	0x3000 0301	
	…	…	
	Function_Key_254	0x3000 03FE	
	Function_Key_255	0x3000 03FF	Reserved
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

Category	Mnemonic	KeySymValue	Description
	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	

1 Table 43: Key Event Mapping

APPENDIX C – LANGUAGE SETS

Basic Set Latin-1

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

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

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