ETSI TS 103 544-4 V1.3.1 (2019-10)

Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 4: Device Attestation Protocol (DAP)

CAUTION

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

Reference
RTS/ITS-98-4

Keywords
interface, ITS, PAS, smartphone

ETSI

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

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

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

Important notice

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

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at www.etsi.org/deliver.

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

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

Copyright Notification

Contents

Intellectual Property Rights

Essential patents

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

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

Trademarks

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

Foreword

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

The present document is part 4 of a multi-part deliverable. Full details of the entire series can be found in part 1 [i.1].

Modal verbs terminology

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

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

1 Scope

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

The term "device attestation" in this context refers to the MirrorLink client verifying that the MirrorLink server is from a compliant manufacturer and running approved software. The attestation will be based on standard X.509 certificates [2] and attestation mechanisms defined by Trusted Computing Group®.

2 References

2.1 Normative references

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

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

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

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

[1] W3C Recommendation 11 April 2013: "XML Signature Syntax and Processing Version 1.1".

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

[2] IETF RFC 5280: "Internet X.509 Public Key Infrastructure Certificate", May 2008.

NOTE: Available at http://tools.ietf.org/html/rfc5280.

[3] IETF RFC 3279: "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", April 2002.

NOTE: Available at http://tools.ietf.org/html/rfc3279.

[4] Trusted Platform Module (TPM) specifications, Version 1.2.

NOTE: Available at http://www.trustedcomputinggroup.org/resources/tpm-main-specification.

[5] TCG Mobile Trusted Module Specification, Version 1.0, April 2010.

NOTE: Available at https://trustedcomputinggroup.org/resource/mobile-phone-work-group-mobile-trusted-module-specification/.

[6] Recommendation ITU-T X.690 (08/2015): "Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)".

[7] ETSI TS 103 544-12 (V1.3.1): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 12: UPnP Server Device".

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

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

[10] ETSI TS 103 544-2 (V1.3.1): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 2: Virtual Network Computing (VNC) based Display and Control".

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

[12] Trusted Computing Group, "Credentials Profiles Specification 1.1", May 2007.

NOTE: Available at http://www.trustedcomputinggroup.org/resources/infrastructure-work-group-tcg-credential-profiles-specification.

[13] ETSI TS 103 544-5 (V1.3.1): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 5: Common Data Bus (CDB)".

[14] ETSI TS 103 544-17 (V1.3.1): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 17: MirrorLink over Wi-Fi Display (WFD)".

[15] ETSI TS 103 544-21 (V1.3.1): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 21: High Speed Media Link (HSML)".

2.2 Informative references

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

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

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

[i.1] ETSI TS 103 544-1 (V1.3.1): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 1: Connectivity".

3 Definition of terms, symbols and abbreviations

3.1 Terms

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

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

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

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

3.2 Symbols

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

PK_A	Public key of device A.
PK_{CCC}	Public key of the CCC root CA.
PK_{CTS}	Public key of the CTS root CA (for testing purpose).
PK_{CTSD}	Public key of the CTS Device (for testing purpose).
PK_{CTSM}	Public key of the CTS Manufacturer CA (for testing purpose).
PK_{SD}	Public key of a Server Device.

PK_{SM}	Public key of a Server Manufacturer CA.
SK_A	Private key of device A.
SK_B	Private key of device B.
SK_{CCC}	Private key of the CCC root CA.
SK_{CTS}	Private key of the CTS root CA (for testing purpose).
SK_{CTSD}	Private key of the CTS Device (for testing purpose).
SK_{CTSM}	Private key of the CTS Manufacturer CA (for testing purpose).
SK_{SD}	Private key of a Server Device.
SK_{SM}	Private key of a Server Manufacturer CA.

3.3 Abbreviations

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

ASN.1	Abstract Syntax Notation One
CA	Certification Authority
CCC	Car Connectivity Consortium
CDB	Common Data Bus
CTS	Conformance Test System
DAP	Device Attestation Protocol
DER	Distinguished Encoding Rules
HSML	High-Spead Media Link
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
ML	MirrorLink
MTM	Mobile Trusted Module
OID	Object IDentifier
OS	Operating System
PCR	Platform Configuration Register
PKCS	Public Key Cryptography Standards
RFB	Remote Framebuffer
RSA	Rivest-Shamir-Adleman
RTP	Real-time Transport Protocol
SHA	Secure Hash Algorithm
SOAP	Simple Object Access Protocol
TCP	Transmission Control Protocol
TPM	Trusted Platform Module
UDP	User Datagram Protocol
UINT	Unsigned INTeger
UPnP	Universal Plug and Play
URL	Universal Resource Locator
VNC	Virtual Network Computing
WFD	Wi-Fi Display
XML	eXtensible Markup Language
XSD	XML Schema Definition

4 Managing a DAP Session

4.1 Bindings

4.1.1 TCP Binding

4.1.1.1 Identifying DAP Server

The identification of the DAP server is described in [8].

4.1.1.2 Device Attestation Launch

The DAP server start-up is facilitated via the UPnP *TmApplicationServer*:1 service *LaunchApplication* action, as defined in [8]. The *LaunchApplication* action shall return with a URL to the DAP server.

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

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

a) DAP server shall listen for the DAP client to make TCP connection at the provided URL for at least 10 s.

b) DAP client shall make a TCP connection to the provided URL.

c) DAP server and client shall start DAP according to the steps defined in Clause 5.

4.1.2 Intentionally Terminating the DAP Session

The DAP server shall not intentionally terminate a DAP session.

The DAP client shall intentionally terminate a DAP session, using the following steps:

1) UPnP Control Point uses TmApplicationServer:1 service's *TerminateApplication* SOAP action to send termination request.

2) DAP client shall disconnect the TCP connection.

3) DAP server should disconnect the TCP connection on detection of the client TCP disconnect or 5 s after responding to the *TerminateApplication* SOAP action, whichever comes first.

The DAP client shall wait for any outstanding Device Attestation Response messages, for at least 10 s, prior to terminating the DAP session. The DAP Server shall provide a DAP response to any DAP request within 10 s.

4.1.3 Unintentionally Terminating the DAP Session

Unintentional termination of the DAP session may happen at any time due to error conditions. In the case of unintentional termination of the DAP session, the respective DAP server or client shall disconnect the TCP connection. The respective counterpart should disconnect as well.

If the MirrorLink Client decides to re-establish the DAP session, it shall follow the steps given in clause 4.1.1.2.

To avoid the DAP server or client being 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 variant), allowing the operating system to reuse a port address, even it is currently in the TIME-WAIT state or the DAP server should use a different, unaffected port number.

4.2 Other Bindings

Besides TCP/IP, it will be possible to run MirrorLink Device Attestation Protocol on top of other protocols like Bluetooth RFCOMM, but how to discover and establish connection for such configuration is outside the scope of the present document.

4.3 Testing Considerations

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

The MirrorLink Client shall have a mechanism to allow a test engineer to launch a DAP session (either automatically or manually).

For DAP testing purposes, the MirrorLink Client shall use a CTS root certificate to validate responses from the CTS Server. This CTS root certificate shall be decoupled from the regular CCC root certificate used during production.

The CTS root certificate shall be accepted from the MirrorLink Client in test setup only. The CTS public key (DER encoded) and the 32-byte SHA-256 hash of the CTS public key (Base64 encoded) are provided separately.

The DAP trust chain, for testing purposes is shown in Figure 1.

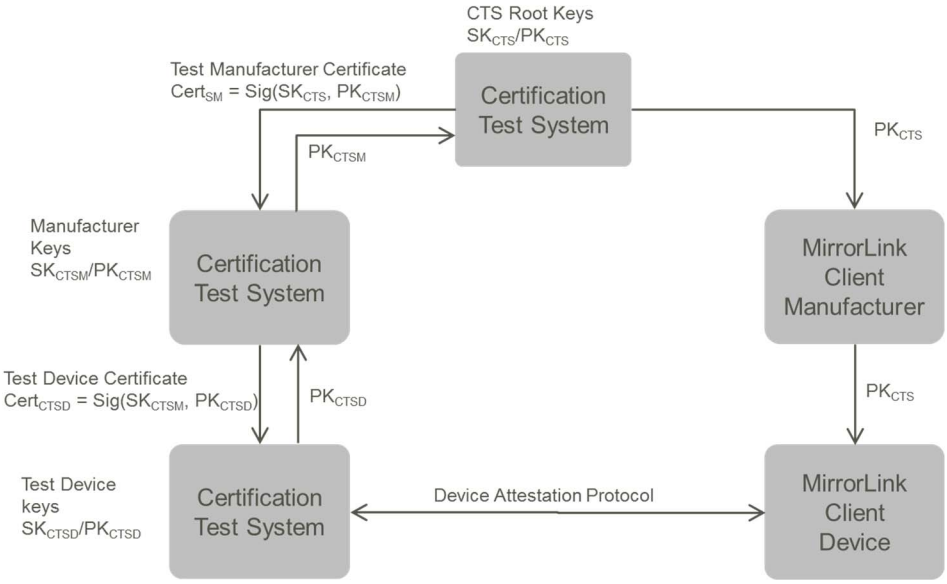

Figure 1: Device Attestation Certification Infrastructure - Testing Only

The MirrorLink Client shall not accept and request the CTS root certificate outside DAP testing and certification.

For testing MirrorLink Servers, the CTS shall only accept legitimate certificates signed by the CCC's Root Certificate.

5 Device Attestation Protocol

The prerequisite of successful Device Attestation Protocol run is that the MirrorLink server has a X.509 device certificate (with Extended Key Usage *tcg-kp-AIKCertificate* OID 2.23.133.8.3 as specified in clause 3.5 of [12]) for its device key pair from the server device manufacturer. The MirrorLink Client shall not expect other X.509 certificate extensions, mandated e.g. in clause 3.4 or 3.5 of [12]. Additionally, the server shall have one X.509 manufacturer certificate signed from the CCC DAP management system. The server device's private key(s) shall be stored securely. The secure storage is manufacturer specific and may use:

1) hardware-based Mobile Trusted Module (MTM) [5] implementation or equivalent;
2) storage on OS, which integrity has been verified with hardware-assisted secure boot process; or
3) storage on OS alone.

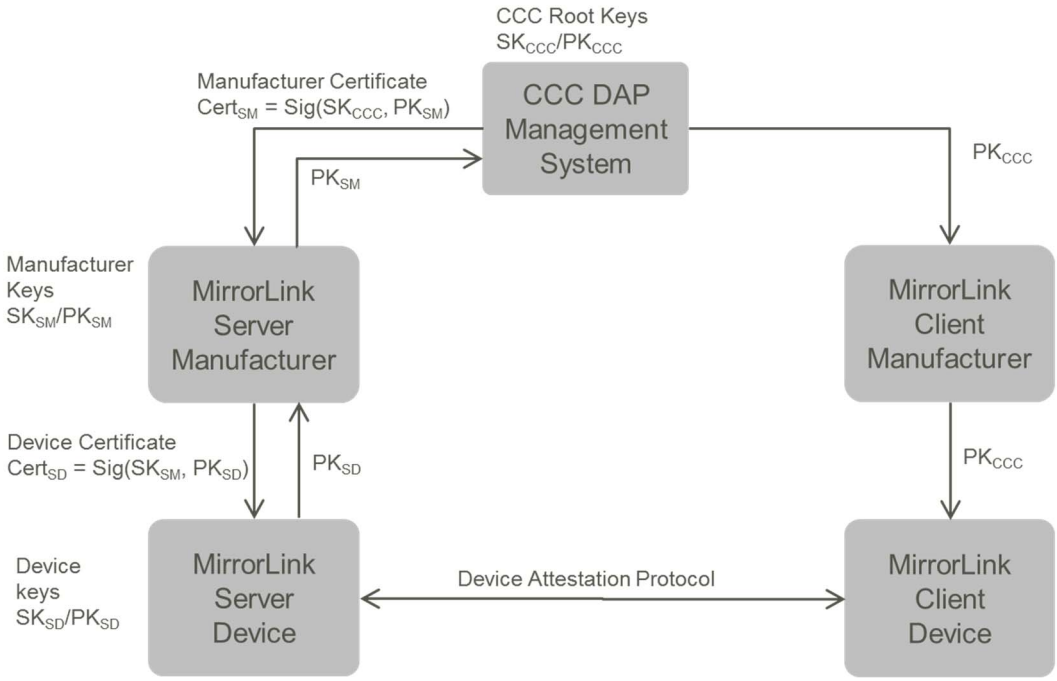

Figure 2: Device Attestation Certification Infrastructure

MirrorLink assumes pre-established trust relationships and security associations between the MirrorLink server device manufacturers and MirrorLink client device manufacturers via a central CCC controlled DAP Management System which extends to both client and server devices. This is achieved using a standard X.509 certificate chain.

The key pair SK_A/PK_A, as shown in Figure 2 consists of the private key SK_A and the public key PK_A. The certificate $Cert_A = Sig(SK_B, PK_A)$ is an X.509 public key certificate with subject public key PK_A and signed with private key SK_B (i.e. the certificate issuer is B).

After the MirrorLink Server device manufacturers have been certified from the central CCC DAP management system $(Cert_{SM} = Sig(SK_{CCC}, PK_{SM}))$, they can certify individual devices they produce $(Cert_{SD} = Sig(SK_{SM}, PK_{SD}))$. Again, MirrorLink specifies using standard X.509 certificates. This certification will typically take place during device manufacturing time, but some device manufacturers may have proprietary mechanisms and the device cert may be bootstrapped during first boot. This operation is done only once per each device.

Using the device certificate, the MirrorLink Server device can authenticate/attest itself to the MirrorLink Client device. For this MirrorLink specifies using Device Attestation Protocol (DAP). This process (DAP) will be run for each connection from a MirrorLink Server device to the MirrorLink Client device. Both the device and the manufacturer certificates are included in the DAP message exchange. Using this chain of certificates, the MirrorLink Client device can verify at runtime that the MirrorLink Server device is a genuine/compliant device from an authorized MirrorLink Server device manufacturer. The MirrorLink Client shall have the public key PK_{CCC} available, to be able to validate the MirrorLink Server device manufacturer certificate.

An overview of device and software attestation protocol is shown in Figure 3 below. The protocol is two-flow protocol: MirrorLink client sends *attestationRequest* message and MirrorLink server replies with *attestationResponse* message. Both of these messages are XML formatted.

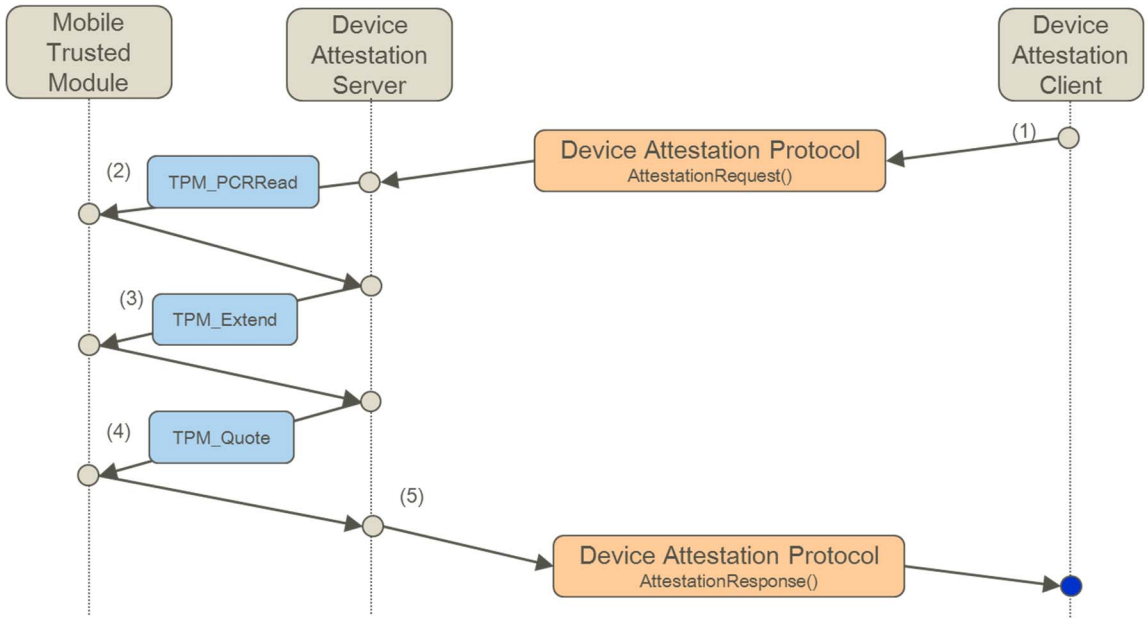

Figure 3: Device and software attestation protocol overview

In more detail, the Device Attestation Protocol consists of following steps:

1) Device Attestation Client first picks random nonce and sends that together with identifier of the requested attested component(s) to Device Attestation Server as part of *AttestationRequest* message.

2) Device Attestation Server is a trusted software component on the MirrorLink server device. It first measures the requested component(s).

NOTE: Measurement of a software component is a platform specific mechanism of identifying a software component on the device. In practice, this could involve verification of a unique software component identifier provided by the underlying operating system security framework, or calculation of a hash over the software component binary executable, or other equivalent mechanisms.

If the measurement does not match expected value, the Device Attestation Server sends a DAP *AttestationResponse* with error code 5. If the measurement matches expected value, the Device Attestation Server proceeds with the next step of the attestation protocol as shown in Figure 3. The Device Attestation Server first reads current value of Platform Configuration Register (PCR) 10 from Mobile Trusted Module using *TPM_PCRRead* command [4]. Device Attestation Server saves the current PCR value for later use. (Device Attestation Server protection and software component measurement mechanisms are platform and implementation specific and intentionally left out of the present document).

3) Device Attestation Server extends *attestationEvidence* to PCR 10 using *TPM_Extend* command [4]. *attestationEvidence* = *componentID* || *URL* || *SHA1*(*applicationPublicKey*), where || denotes to concatenation (for more details on these data elements, see descriptions of *attestationRequest* and *attestationResponse* below).

4) Device Attestation Server performs *TPM_Quote* command [4] on PCR 10 using the nonce as an input. This operation signs the PCR structure using a certified device key. The *TPM_Quote* operation returns *TPM_PCR_COMPOSITE* structure using which the server constructs matching *TPM_QUOTE_INFO* structure.

5) The saved PCR value, the resulting *TPM_Quote* signature, URL, the device certificate, and the device manufacturer certificate are sent to Device Attestation Client as part of *AttestationResponse*. Device Attestation Client verifies the device manufacturer certificate (using pre-installed trust root), verifies device certificate using verified device manufacturer certificate, and finally verifies the attestation signature with respect to the *TPM_QUOTE_INFO* using the verified device certificate.

Implementation considerations:

It is the responsibility of the MirrorLink Server that the usage of certified device key is restricted to authorized components only, such as hardware-based MTM implementation or trusted software component. The exact protection mechanisms are platform and implementation specific.

The elements of *attestationRequest* XML message shall be as defined in Table 1. The XSD schema of the XML shall be as given in Annex A.

Table 1: Device Attestation - *attestationRequest* elements

Element	Description	Parent	Availability
attestationRequest	Main container of attestation request	None	Optional
version	MirrorLink Version information	attestation Request	Mandatory
majorVersion	Major version number	version	Mandatory
minorVersion	Minor version number	version	Mandatory
trustRoot	Identifier of the trust root used by MirrorLink client for attestation verification. 32-byte SHA-256 hash of Certificate Authority's root public key Base64 encoded from SubjectPublicKeyInfo DER format [1]	attestation Request	Mandatory
nonce	20-byte random number Base64-encoded [1]	attestation Request	Mandatory
componentID	Identifier of the software component to be attested. Possible values are described in Table 2	attestation Request	Mandatory

Device Attestation Protocol can be used to enable the following attestations:

- Attest that the connected MirrorLink Server device is manufactured from a trusted manufacturer.
- Attest that software components on the connected MirrorLink Server device are original (i.e. certified).

The components, which can be attested, shall be as defined in Table 2 with their component IDs. The Protocol ID is used as part of the attestation's response URL binding.

Table 2: Device Attestation - Component List

componentID	Description of what functionality is attested	Protocol ID
TerminalMode:VNC-Server	VNC server functionality as specified in [10].	VNC
TerminalMode:UPnP-Server	UPnP *TmServerDevice* server services as specified in [7], [8] and [9]. The attestation includes the service advertisements, using UDP broadcasts and the SOAP and eventing mechanisms based on HTTP.	HTTP
TerminalMode:RTP-Server	RTP-Server as specified in [11].	RTP
TerminalMode:RTP-Client	RTP-Client as specified in [11].	RTP
MirrorLink:CDB-Endpoint	CDB Endpoint, as specified in [13] (≥ MirrorLink 1.1).	CDB
MirrorLink:WFD:RTSP	WFD Source, as specified in [14] (≥ MirrorLink 1.2).	WFD
MirrorLink:HSML	HSML Source as specified in [15] (≥ MirrorLink 1.2).	VNC

componentID	Description of what functionality is attested	Protocol ID
MirrorLink:Device	Device is certified and manufactured from a CCC member (≥ MirrorLink 1.1).	-
*	Wildcard. All components, which can be attested from the MirrorLink Server, shall be attested. In this case the MirrorLink server shall reply with a single *attestationResponse* message, which includes the attestation elements of all attested components.	-

The MirrorLink Server shall support attesting of the following components (DAP Minimum Set):

- `"MirrorLink:Device"`
- `"TerminalMode:UPnP-Server"`, including the *URL* and the *applicationPublicKey*

The MirrorLink Client shall accept a MirrorLink Server, implementing the above listed DAP Minimum Set, in Park and Drive Mode.

The MirrorLink Server should provide support for attesting other components. Components, the MirrorLink Server is not able to attest, shall not be included in the *attestationResponse*.

The MirrorLink Client shall execute DAP, when connected to a MirrorLink 1.1 or beyond Server, for at least the DAP Minimum Set, as defined above. The execution of DAP may be deferred, to satisfy initial connection setup latency requirements. Deferral shall not exceed 1 min.

Implementation Note:

MirrorLink 1.1 and 1.2 Client may not execute DAP, or may execute it only in Drive Mode.

Proprietary components shall be identified with a *domainName*:*componentID*, where *domainName* follows the Java namespace convention (e.g. `com.daimler`). The MirrorLink server shall ignore the attestation of proprietary *componentIDs*, which are not supported by it.

Below is an example of *attestationRequest* message (from a MirrorLink 1.3 Client):

```
<attestationRequest>
    <version>
        <majorVersion>1</majorVersion>
        <minorVersion>3</minorVersion>
    </version>
    <trustRoot>dbR5…dT5S3=</trustRoot>
    <nonce>7Thg34saHd5...4hkjd=</nonce>
    <componentID>TerminalMode:VNC-Server</componentID>
</attestationRequest>
```

The elements of *attestationResponse* XML message shall be as defined in Table 3. The XSD schema of the XML shall be as given in Annex A:

Table 3: Device Attestation - *attestationResponse* Elements

Element	Description	Parent	Availability
attestation Response	Main container of attestation response.	None	Optional
version	MirrorLink Version information.	attestation Response	Mandatory
major Version	Major version number.	version	Mandatory
minor Version	Minor version number.	version	Mandatory
result	Indicates success/failure of attestation operation. Possible values are defined in Table 5.	attestation Response	Optional
sizeOfSelect	UINT 16 value of the sizeOfSelect as used by the ML Server in the TPM_PCR_SELECTION structure when calling TPM_Quote. (**Default**: 3).	attestation Response	Optional

Element	Description	Parent	Availability
attestation*	Contains attestation of one component. Mandatory only in case of successful attestation (result == 0).	attestation Response	Mandatory
componentID	Identifier of the attested component. Possible values are listed in Table 2.	attestation	Mandatory
oldValue	The old value of the PCR 10 reserved for device and software attestation use. 20-byte binary value Base64-encoded [1].	attestation	Mandatory
quoteInfo	TPM_QUOTE_INFO structure (as specified in [4]) over which the quoteSignature is calculated. Contains TPM_COMPOSITE_HASH value derived from the content of PCR 10. Base64-encoded [1].	attestation	Mandatory
quote Signature	RSA PKCS#1 v1.5 formatted signature produced by TPM_Quote command as specified in [4]. 256-byte binary value Base64-encoded [1].	attestation	Mandatory
URL	URL that defines the Protocol ID (according Table 2), IP address and port number that the attested software component is currently holding, according the following format: [ProtocolID]://[IP]:[Port] shall be left empty for MirrorLink:Device component identifier. Multiple URLs of the same component shall be added as separate attestation elements.	attestation	Mandatory
application PublicKey	Public part of key pair that the attested application may use to authenticate (e.g. sign) transferred data. (The private part of this key pair should be accessible only by the attested application. The mechanism used to protect the private key depends on the platform of server device.) The key pair shall be 2048 bit RSA key and the public part shall be Base64 encoded [1] from X.509 SubjectPublicKeyInfo DER format [2].	attestation	Optional
device Certificate	X.509v3 [2] certificate issued by the MirrorLink server device manufacturers. The certificate contains the public part of the 2048-bit RSA device key with SHA-256 or SHA-512. The certificate may have variable length. The certificate is Base64 encoded from ASN.1 DER format [6]. Mandatory only in case of successful attestation (result == 0).	attestation Response	Mandatory
manufacturer Certificate*	A (chain of) X.509v3 [2] certificate(s) issued for the MirrorLink server manufacturer by the Certificate Authority. The certificate contains the public part of the 2048-bit or 4096-bit RSA manufacturer key with SHA-512. The certificate(s) may have variable length. The certificate(s) are Base64 encoded from DER format. Mandatory only in case of successful attestation (result == 0).	attestation Response	Mandatory

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

The usage of the application public key shall be as defined in the following Table 4.

Table 4: Device Attestation - Application Public Key

componentID	Description of how the Application Public Key is used
TerminalMode:VNC-Server	Used for VNC Content Attestation [10]
TerminalMode:UPnP-Server	Used for UPnP XML signature validation [7], [8] MANDATORY
TerminalMode:RTP-Server	Not used; reserved for future use
TerminalMode:RTP-Client	Not used; reserved for future use
MirrorLink:CDB-Endpoint	Used for CDB payload encryption [13]
MirrorLink:WFD:RTSP	Not used; reserved for future use
MirrorLink:HSML	Not used; reserved for future use
MirrorLink:Device	Not used; reserved for future use

The MirrorLink Client shall verify the X.509 trust chain, using only RSA with SHA-2. Algorithms are defined in [3].

The MirrorLink Client shall validate the validity time of the certificates.

In case the MirrorLink Client does not have access to time, it shall check the certificate's expiration date. The certificates shall not have expired before the MirrorLink Client has been manufactured or the MirrorLink Client stack has been build (shall not be earlier than 6 months prior to seeking device certification for the MirrorLink Client).

The MirrorLink Server shall not use a *componentID* value equal to the wildcard `"*"` within the *attestationResponse* message.

The MirrorLink Server shall not include more than 3 entries into the manufacturer certificate chain. To simplify the validation for the MirrorLink Client, the MirrorLink Server shall include the manufacturer certificates in trust chain order, i.e. the certificate from the CA, which signed the device certificate, shall be first and the certificate, which was signed by the CCC root CA shall be last.

An example with 3 entries is shown below.

```
<deviceCertificate>
    Device Certificate, signed by manufacturer Sub-Sub-CA
</deviceCertificate>
<manufacturerCertificate>
    Manufacturer Sub-Sub-CA Certificate, signed by manufacturer Sub-CA
</manufacturerCertificate>
<manufacturerCertificate>
    Manufacturer Sub-CA Certificate, signed by manufacturer CA
</manufacturerCertificate>
<manufacturerCertificate>
    Manufacturer CA certificate, signed by CCC root CA
</manufacturerCertificate>
```

An example with 1 entry is shown below.

```
<deviceCertificate>
    Device Certificate, signed by manufacturer CA
</deviceCertificate>
<manufacturerCertificate>
    Manufacturer CA certificate, signed by CCC root CA
</manufacturerCertificate>
```

Possible result values indicating success/failure of the attestation operation are given in Table 5. In case the MirrorLink Client request attestation of an individual component, which is not available for attestation, the MirrorLink Server shall respond with the `"Component not existing"` response.

Table 5: Device Attestation - Possible Attestation Result Values

Result	Description of result value
0	Successful attestation (**default**)
1	Component not existing or attestation not available
2	Error in attestation: Version not supported
3	Error in attestation: unknown trust root
4	Error in attestation: unknown component ID
5	Error in attestation: Attestation failed

Below is an example of *attestationResponse* message (for a MirrorLink 1.3 Server, responding to a MirrorLink 1.3 Client):

```
<attestationResponse>
    <version>
        <majorVersion>1</majorVersion>
        <minorVersion>3</minorVersion>
    </version>
    <result>0</result>
    <attestation>
        <componentID>TerminalMode:VNC-Server</componentID>
        <oldvalue>jlFGh...kj=</oldValue>
        <quoteInfo>kDal2d33...26sE56jJsa</quoteInfo>
        <quoteSignature>IL7h9j9...4J3234Kg=</quoteSignature>
        <URL>VNC://192.168.64.1:5500</URL>
    </attestation>
    <deviceCertificate>gTsd7d3...763rJKh=</deviceCertificate>
    <manufacturerCertificate>6sbk5..7d4dkH= </manufacturerCertificate>
</attestationResponse>
```

The MirrorLink Client shall not display any MirrorLink Server content in drive-mode, in case the attestation failed, the DAP response cannot be validated, or the DAP trust chain cannot be validated back to the Certificate Authority's root. The MirrorLink Client shall provide the consumer with a notification.

The MirrorLink Client should terminate the MirrorLink session in this case. Otherwise, in case the attestation failed, the DAP response cannot be validated, or the DAP trust chain cannot be validated back to the Certificate Authority's root, the MirrorLink Client shall treat the server device and all its provided content as uncertified. The MirrorLink Client shall provide the consumer with a notification.

The MirrorLink Client shall use a version number, which is equal or smaller than the MirrorLink Server's version number, as advertised in the launched DAP endpoint's Remote Access Protocol Format entry.

The MirrorLink Server shall return a version number, which is equal or smaller than the MirrorLink Client's version number. That means that the MirrorLink Server shall not include components into a DAP response to a wildcard "*" DAP request, which have not been specified for the respective MirrorLink Client version, as given below:

- A MirrorLink 1.1 Server connected to a MirrorLink 1.0 Client shall not include any MirrorLink:Device, MirrorLink:CDB-Endpoint components.
- A MirrorLink 1.2 or 1.3 Server connected to a MirrorLink 1.0 Client shall not include any MirrorLink:Device, MirrorLink:CDB-Endpoint, MirrorLink:HSML, MirrorLink:WFD:RTSP components.
- A MirrorLink 1.2 or 1.3 Server connected to a MirrorLink 1.1 Client shall not include any MirrorLink:HSML, MirrorLink:WFD:RTSP components.

It shall be noted though, that a MirrorLink Server shall correctly respond to a DAP request for any individual component (i.e. excluding the wildcard), even if that component is not specified for the respective MirrorLink Client version (but is specified within the respective MirrorLink Server version).

To verify *quoteSignature* a MirrorLink client should construct a *TPM_PCR_COMPOSITE* structure and validate the *TPM_Quote* signature with respect to that. The *TPM_PCR_COMPOSITE* structure contains a *TPM_PCR_SELECTION* structure in which only PCR 10 is set (bit field starting with 0x00 0x04 and padded with 0x00 up to a length in bytes of *sizeOfSelect*). For more details on these structures see TPM Structures specification from [3]. The MirrorLink client should perform following steps:

1) Calculate hash H1 as SHA1(applicationPublicKey) if attestationResponse message included applicationPublicKey element.

2) Calculate hash H2 as SHA1(oldValue || SHA1(componentID || URL || H1)). Include H1 if attestationResponse message included applicationPublicKey element.

3) Create TPM_PCR_COMPOSITE structure C:

   a) Set C->select->sizeOfSelect to sizeOfSelect (UINT16 value)

   b) Set C->select->pcrSelect[0] to 0x00

   c) Set C->select->pcrSelect[1] to 0x04

   d) Pad C->select->pcrSelect with 0x00 (until size of C->select->pcrSelect is *sizeOfSelect*)

   e) Set C->valueSize to 20 (UINT32 value)

   f) Set C->pcrValue to H2

4) Calculate hash *H3* as SHA1(C).

5) Verify that *digestValue* in *quoteInfo* equals to H3.

6) Verify that *externalData* in *quoteInfo* equals to nonce in *AttestationRequest.*

7) Verify that received quoteSignature is valid RSA-SHA1 PKCS#1 v1.5 signature over received *quoteInfo* using public part of device key extracted from *deviceCertificate*. (This means that *quoteSignature* should valid signature over SHA1(*quoteInfo*)).

The client shall verify that the server device certificate has *tcg-kp-AIKCertificate* Extended Key Usage as specified in [12]. Otherwise the attestation shall not be accepted.

If device and software attestation is mandated by the MirrorLink client, it should attest all software components on the MirrorLink server before presenting content received from these components to the user.

When a software component on the MirrorLink server is attested, the client should check that it has active (TCP) connection with the attested software component that matches the attested IP address and port number. Once the active (TCP) connection breaks the client should run the attestation protocol again for the same component (if mandated by the client).

NOTE: When device and software attestation protocol is run over networked connection, such as protected Wi-Fi, an attacker *within* the same network may be able to masquerade as the attested device. Performing such an attack requires that the attacker is able to (1) spoof its IP address and (2) manipulate TCP connection parameters (such as sequence numbers). Both are possible with right kind of equipment, but the user will need to have intentionally added the malicious device into the protected Wi-Fi network. In such a case, it is recommended to use the application specific key (that was bound to attestation of each component) to authenticate (e.g. sign on server and verify on client) all or a subset of the communication.

It is recommended to terminate the Device Attestation Protocol, using the appropriate *TerminateApplication*() SOAP action after all components have been successfully attested. The Device Application Protocol can be launched again at any point in time.

Annex A (normative): XSD Schema

DAP attestationRequest XSD Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="urn:schemas-upnp-org:dapserver" xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified" id="attestationRequest">
<xs:element name="attestationRequest">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="version" minOccurs="1" maxOccurs="1">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="majorVersion" type="xs:nonNegativeInteger" minOccurs="1" maxOccurs="1"/>
      <xs:element name="minorVersion" type="xs:nonNegativeInteger" minOccurs="1" maxOccurs="1"/>
      <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
     </xs:sequence>
     <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
   </xs:element>
   <xs:element name="trustRoot" type="xs:string" minOccurs="1" maxOccurs="1"/>
   <xs:element name="nonce" type="xs:string" minOccurs="1" maxOccurs="1"/>
   <xs:element name="componentID" type="xs:string" minOccurs="1" maxOccurs="1"/>
   <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
 </xs:complexType>
</xs:element>
</xs:schema>
```

DAP attestationResponse XSD Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="urn:schemas-upnp-org:dapserver" xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified" id="attestationResponse">
<xs:element name="attestationResponse">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="version" minOccurs="1" maxOccurs="1">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="majorVersion" type="xs:nonNegativeInteger" minOccurs="1" maxOccurs="1"/>
      <xs:element name="minorVersion" type="xs:nonNegativeInteger" minOccurs="1" maxOccurs="1"/>
      <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
     </xs:sequence>
     <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
   </xs:element>
   <xs:element name="result" minOccurs="0" maxOccurs="1" default="0"/>
   <xs:element name="sizeOfSelect" minOccurs="0" maxOccurs="1" default="3">
    <xs:simpleType>
     <xs:restriction base="xs:unsignedShort">
      <xs:minInclusive value="2"/>
     </xs:restriction>
    </xs:simpleType>
   </xs:element>
   <xs:element name="attestation" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="componentID" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="oldValue" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="quoteInfo" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="quoteSignature" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="URL" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="applicationPublicKey" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
     </xs:sequence>
     <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
   </xs:element>
   <xs:element name="deviceCertificate" type="xs:string" minOccurs="0" maxOccurs="1"/>
   <xs:element name="manufacturerCertificate" type="xs:string" minOccurs="0" maxOccurs="3"/>
```

```
   <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
 </xs:complexType>
</xs:element>
</xs:schema>
```

Annex B (informative): Authors and Contributors

The following people have contributed to the present document:

Rapporteur:	Dr. Jörg Brakensiek, E-Qualus (for Car Connectivity Consortium LLC)
Other contributors:	Mark Beckmann, Volkswagen AG
	Matthias Benesch, Daimler AG
	Raja Bose, Nokia Corporation
	Dennis Fernahl, Carmeq (for Volkswagen AG)
	Kari Kostiainen, Nokia Corporation
	Martin Lehner, Jambit
	N. Asokan, Nokia Corporation
	Keun-Young Park, Nokia Corporation
	Michael Wolf, Daimler AG

History

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