ETSI TS 103 544-5 V1.3.1 (2019-10)

Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 5: Common Data Bus (CDB)

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-98-5

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 5 of a multi-part deliverable. Full details of the entire series can be found in part 1 [i.1].

Modal verbs terminology

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

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

1 Scope

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

The Common Data Bus (CDB) is a simple, low-bandwidth shared bus, which allows exchanging data between two CDB endpoints, residing in the MirrorLink Server and Client. The Common Data Bus is fully symmetrical, i.e. services can be provided on both endpoints independently from each other.

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

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

Void.

3.2 Symbols

Void.

3.3 Abbreviations

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

API	Application Programming Interface
CCC	Car Connectivity Consortium
CDB	Common Data Bus
DAP	Device Attestation Protocol
IP	Internet Protocol
TCP	Transmission Control Protocol
UPnP	Universal Plug-and-Play
URL	Universal Resource Locator
VNC	Virtual Networking Computing

4 Common Data Bus Architecture

4.1 General Architecture

The Common Data Bus (CDB) is a simple, low-bandwidth shared bus, which allows exchanging data between two CDB endpoints, residing in the MirrorLink Server and Client. The Common Data Bus is fully symmetrical, i.e. services can be provided on both endpoints independently from each other.

A CDB endpoint can host a CDB Sink endpoint and a CDB Source endpoint. CDB Sink endpoints are subscribing to data services provided from CDB Source endpoints. A CDB source endpoint can provide multiple data services from data sources. A CDB sink endpoint can deliver data from multiple data sources to multiple data sinks. The endpoints are responsible for marshalling and de-marshalling of all the data from multiple applications passing through the common data bus. A CDB data sink subscribes to a service, provided from a data source service. For simplicity, the following figure shows two CDB services, which one data source and data sink each.

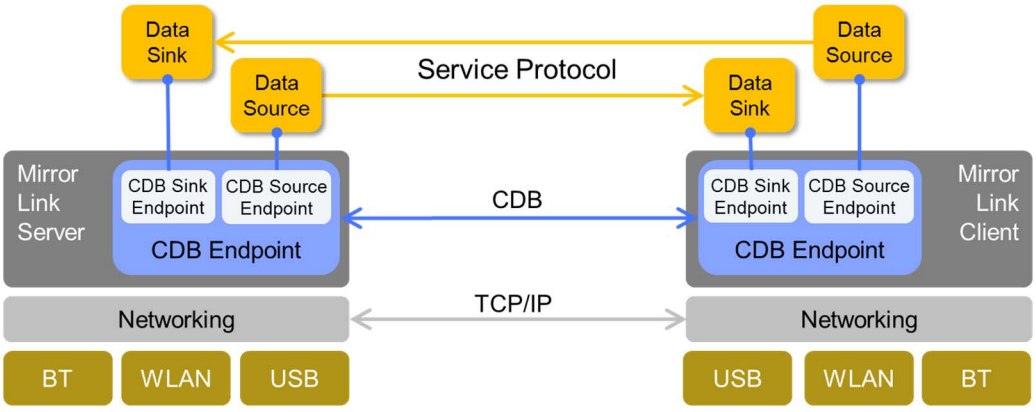

Figure 1: Common Data Bus Architecture with TCP binding

4.2 Bindings

4.2.1 TCP Binding

4.2.1.1 Launching the Common Data Bus

The MirrorLink Server, providing Common Data Bus functionality, shall include the Common Data Bus into its application listing as specified in [1]. The MirrorLink Client shall identify the Common Data Bus from the application listing as specified in [1].

The Common Data Bus shall be started from the MirrorLink Client, using the UPnP *TmApplicationServer*:1 service *LaunchApplication* action [1]. The MirrorLink Client's CDB endpoint shall open a TCP connection to the Server's CDB endpoint, using the returned URL from the *LaunchApplication* action.

The MirrorLink Client should launch the MirrorLink Server's CDB endpoint not later than 10s after receiving the *first A_ARG_TYPE_AppList* response from the MirrorLink Server.

The MirrorLink Client shall have launched the MirrorLink Server's CDB prior starting the first VNC based remote application.

4.2.1.2 Intentionally Terminating the Common Data Bus

The MirrorLink Client and Server can terminate the Server's Common Data Bus endpoint anytime, using the CDB *ByeBye* message and the UPnP *TmApplicationServer:1* services, as defined in [1].

The CDB endpoint in the MirrorLink Client shall use the following sequence to terminate the CDB operation:

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

2) Server CDB endpoint shall respond with a CDB *ByeBye* message.

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

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

Client CDB endpoint should send a UPnP *TmApplicationServer*:1 service *TerminateApplication* action for the server CDB endpoint.

The CDB endpoint in the MirrorLink Server shall use the following sequence to terminate the CDB operation:

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

2) Client CDB endpoint shall disconnect the TCP connection.

3) Server CDB endpoint shall signal the CDB endpoint's termination to the Client, if it has subscribed to the *TmApplicationServer:1 AppStatusUpdate* event.

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

If the CDB is terminated prior to the establishment of the TCP connection, steps 1, 2 and 4 shall be omitted.

4.2.1.3 Unintentionally Terminating the CDB Session

Unintentional termination of the CDB session may happen any time in case of error conditions. In this case the respective CDB Server or Client endpoint will disconnect the TCP connection. The respective counterpart should disconnect as well.

If the MirrorLink Client decides to re-establish the CDB session, it shall follow the steps given in clause 4.2.1.1.

To avoid the CDB Server or Client endpoint 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 variants), allowing the operating system to reuse a port address, even it is currently in the TIME-WAIT state or the CDB Server endpoint should use a different, unaffected port.

4.2.2 Other Bindings

Besides TCP/IP, it will be also possible to run MirrorLink Common Data Bus on top of other protocol 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 CDB session (either initiated automatically or manually from the user), whenever the CDB Server endpoint has intentionally terminated the CDB session.

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

5 Message Types and Format

5.1 Message Overview

The Common Data Bus (CDB) defines the following messages, which are specified in more detail in the following paragraphs:

- *ServicesRequest*: Requests the list of supported services
- *ServicesSupported*: Provide a list of supported data services
- *StartService*: Request to start a specific service
- *StopService*: Request to stop a specific service
- *ServicePayload*: Deliver service specific payload
- *ServiceResponse*: Response to *StartService*, *StopService*, *ServicePayload*
- *ByeBye*: Terminates the Common Data Bus
- *Ping*: Message to check the connection
- *PingResponse*: Responds to a Ping message

All U16 values are encoded in big endian.

5.2 ServicesRequest

The *ServicesRequest* message is used from the CDB Sink endpoint to request the list of supported services from the CDB Source endpoint.

Table 1: *ServicesRequest* Message

# bytes	Type	Value	Description
2	U16	0xB101	Message-type
2	U16	2	Payload length
1	U8	1	CDB sink endpoint major version
1	U8	1	CDB sink endpoint minor version

After the Common Data Bus connection is initiated between the 2 CDB endpoints, each CDB sink endpoint can send a *ServicesRequest* message to indicate that it is interested in getting the list of services available at the CDB source endpoint and is also interested in getting updates whenever the list changes.

If a CDB endpoint supports CDB sink functionality, the CDB Sink endpoint shall send a *ServicesRequest* message within 5 s after the CDB connection has been established.

The CDB Source endpoint shall respond with a *ServicesSupported* message to the *ServicesRequest* message within 5 s. The CDB Sink endpoint should not send a *ServicesRequest* message, if it does not support service subscription.

In case a response is not received within 5 s, the CDB Sink endpoint shall assume that the CDB Source is not providing any service. No further action is required.

Implementation Note:

The present document defines a CDB sink endpoint version of 1.1. Note that the CDB sink endpoint version is not necessarily identical with the MirrorLink version. The following CDB versions are supported within the different MirrorLink versions:

- MirrorLink 1.1: CDB sink endpoint version 1.1.
- MirrorLink 1.2: CDB sink endpoint version 1.1.
- MirrorLink 1.3: CDB sink endpoint version 1.1.

5.3 ServicesSupported

The *ServicesSupported* message shall be used from the CDB Source endpoint to notify the CDB Sink endpoint about the data services it can provide.

Table 2: *ServicesSupported* Message

# bytes	Type	Value	Description
2	U16	0xB102	Message-type
2	U16	4+M	Payload length
1	U8	1	CDB source endpoint major version
1	U8	1	CDB source endpoint minor version
2	U16	N	Total number of Services
M	Array of U8		Array of service descriptions, as defined in Table 3.

The CDB Source endpoint version shall be equal or smaller than the CDB Sink endpoint's version. Otherwise the CDB Sink endpoint shall send a *ServiceResponse* message with the Error code 0x0209 in response to the received *ServicesSupported* message.

Implementation Note:

The present document defines a CDB source endpoint version of 1.1. Note that the CDB source endpoint version is not necessarily identical with the MirrorLink version. The following CDB versions are supported within the different MirrorLink versions.

- MirrorLink 1.1: CDB source endpoint version 1.1.
- MirrorLink 1.2: CDB source endpoint version 1.1.
- MirrorLink 1.3: CDB source endpoint version 1.1.

Each service is described as given in Table 3.

Table 3: Service Description

# bytes	Type	Value	Description
2	U16		Unique Service ID
1	U8		Major version of service
1	U8		Minor version of service
1	U8	*Bit*	*Service Configuration ('1' enabled, '0' disabled)*
		[0]	Service Encryption Data payload is encrypted with the data provider's session key, exchanged as within the DAP of the Server's CDB endpoint.
		[1]	Service Resource Constraints Only one service can be started, in case multiple entries with same service name, but different version, exist in the services supported list.
1	U8	*Bit*	*Service Access Control ('1' enabled, '0' disabled)*
		[0]	*Unlimited* Data can be provided to or received from any application.
		[1]	*CCC-Certified* Data can be provided to or received from any CCC certified application.
		[2]	*Source-Certified* Data can be provided to or received from any Member certified application.
		[3]	*Service-Certified* Data can be provided to or received from any application, certified from the MirrorLink Client manufacturer to access the data service, as defined in the *serviceList* entry of the application certificate.
1	U8	N	Indicates the length of service name string.
N	Array of U8		Service name string

Each service shall have a service ID, which is unique during a CDB session. Service IDs for services provided from the MirrorLink Server shall have the range of `0x4001` to `0x7FFF`, and service IDs for services from the MirrorLink Client shall have the range of `0x0001` to `0x3FFF`. The most significant bit is reserved for future use and shall be set to 0. The CDB Source endpoint shall not change the Service ID number during a CDB session.

Setting more than one Service Access Control bit, will aggregate (i.e. OR) the access control conditions. The source grants no access to a data service, if all Service Access Control bits in the above table are set to zero (0). The relationship between the Service Access Control layers is shown in Figure 2.

The MirrorLink Client manufacturer shall be identified from the *manufacturer* entry in the UPnP *TmClientProfile:1 SetClientProfile* action. If the Client does not use the service, or the *manufacturer* entry is left empty, the Source- and Service-Certified Service Access Control bits are ignored (i.e. assumed to be zero (0)).

Locale information shall not matter, when deciding, whether an application has access to a data service. Developer application shall not have access to data services, as than those, which have either bit 1 (CCC-certified) or bit 0 (Unlimited) enabled.

Access to data services, which are defined as system services, shall be limited to platform / system services on the MirrorLink Sever device. Access to system services from MirrorLink applications using the Common API shall be denied. A Bluetooth Out-of-Band pairing data service is an example system service.

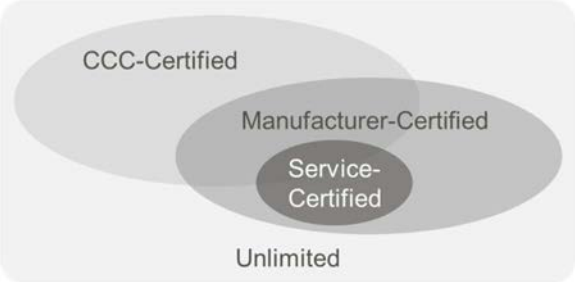

Figure 2: Service Access Control Layers

Service names shall follow the naming convention `domainName.serviceName`, where `domainName` is following the Java namespace convention (e.g. `com.daimler`). MirrorLink specified services shall use `"com.mirrorlink"` as the domain name.

A CDB Source endpoint shall send a *ServicesSupported* message for every *ServicesRequest* message that it receives. However, after the receipt of the first *ServicesRequest* message, a CDB Source endpoint shall send a *ServicesSupported* message, whenever the list of available services changes, even if the other endpoint has not sent any subsequent *ServicesRequest* messages.

The CDB source endpoint shall list only one service with identical service name string and major version. Any CDB services shall be backward compatible with regard to its minor versions.

MirrorLink Applications, that act as a data service source on a MirrorLink Server, shall include the respective service name in the <*serviceList*> element of their application certificate. Only CCC certified applications, and applications certified for the connected MirrorLink Client, shall act as a data service source.

A *ServicesSupported* message shall contain the complete list of services currently available. In case a CDB Source endpoint does not have any services available, it shall respond with a *ServicesSupported* message where the Total number of Services (see Table 2) is set equal to 0.

Data Services may be added in the future, therefore, a CDB Sink shall ignore all services it does not recognize or support. A CDB Sink shall not start a service or otherwise send messages to, which it does not support.

5.4 StartService

The *StartService* message is used from the CDB Sink endpoint to signal to the CDB Source endpoint to start a specific service and commence putting data packets related to that data service on to the common data bus. Table 4 specifies the format of the message.

Table 4: *StartService* Message

# bytes	Type	Value	Description
2	U16	0xB103	Message-type
2	U16	4	Payload length
2	U16		Service Id
1	U8		Major version of service
1	U8		Minor version of service

The service version shall be equal or smaller than the announced service version from the *ServicesSupported* message. The CDB Source endpoint shall respond with a *ServiceResponse* message with the corresponding Service Id within 5 s. Otherwise the CDB Source endpoint shall respond with a *ServiceResponse* message (Error Code set to `0x0010` - Response pending) latest every 5 s until the final response message is available. The CDB Source endpoint shall respond with a *ServiceResponse* message (Error Code set to `0x0201` - Launch failed) if no response can be provided within 2 min.

In case the *ServiceResponse* messages are not received in time, the CDB Sink endpoint shall consider the launch finally failed and shall send a *StopService* message.

5.5 StopService

The *StopService* message is used from the CDB Sink endpoint to signal to the CDB Source endpoint to stop a specific service and stop putting data packets related to that service on to the common data bus. Table 5 specifies the format of the message.

Table 5: *StopService* Message

# bytes	Type	Value	Description
2	U16	0xB104	Message-type
2	U16	2	Payload length
2	U16		Service Id

The CDB Source endpoint shall respond with a *ServiceResponse* message with the corresponding Service Id within 5 s.

In case the *ServiceResponse* message is not received in time, the CDB Sink endpoint shall consider the termination finally failed. No further action is required.

5.6 ServicePayload

The *ServicePayload* message delivers data from one CDB endpoint to the other one for any of the services, which have been started. The payload can have any data format of its own depending on the available services. The specification of the service payload is done within separate specifications and is out of the scope of the present document. Table 6 specifies the format of this message.

Table 6: *ServicePayload* Message

# bytes	Type	Value	Description
2	U16	0xB105	Message-type
2	U16	3+M	Payload length
2	U16		Service Id
1	U8		Service Access Control
M	Array of U8		Payload for the Service

The Service Access Control byte of the *ServicePayload* message shall be identical to the Service Access Control byte from the respective service in the *ServicesSupported* message.

In case the Service Encryption bit is set to `'1'` in the Service Configuration bit of the Service listing, the entire Service Access Control byte and the Service Payload shall be encrypted.

NOTE: The *Service Access Control* byte and *Payload for the Service* entry are concatenated prior encryption. The encrypted data replaces both, the *Service Access Control* byte and *Payload for the Service* entry, in Table 6. The *Payload length* is adjusted accordingly.

The CDB Endpoint on the MirrorLink Client shall use the application specific public key that was bound to attestation of CDB Endpoint Server to encrypt all *ServicePayload* messages sent from the MirrorLink Client to the MirrorLink Server. The encryption is done according to RSAPKCS#1 v1.5 format. Encryption at the CDB Endpoint on the MirrorLink Server is currently not specified in the present document, i.e. encryption of *ServicePayload* messages, sent from the MirrorLink Server to the MirrorLink Client, is currently not possible.

A CDB Endpoint should not launch a CDB data service, which is using encrypted *ServicePayload* messages, as advertised in the CDB Services Supported message, if it is not able to decrypt them.

The service payload length M, as defined in Table 6, shall be equal or smaller than 8 100 bytes. Both MirrorLink Client and Server shall be able to receive a service payload up to this size.

Any service using the Common Data Bus, may split long payloads into smaller service payloads, fulfilling the service payload length requirement. In such case, it is up to each service to fragment and re-assemble those longer packets.

The CDB Sink endpoint shall respond with a *ServiceResponse* message with the corresponding Service Id, if the *ServicePayload* message's Service Id is unknown or if the Service Configuration bits have changed.

5.7 ServiceResponse

The *ServiceResponse* message shall be used from the CDB Source or Sink endpoint to respond to a message send from the CDB Sink or Source endpoint respectively (refer to Table 12 for details). Table 7 specifies the format of this message:

Table 7: *ServiceResponse* Message

# bytes	Type	Value	Description
2	U16	0xB107	Message-type
2	U16	4	Payload length
2	U16		Service Id
2	U16		Response Value

The error types are defined in Table 8.

Table 8: Error Types for *ServiceResponse* message

Response Value		Description
0x0001	Ok - Service started	Service has been successfully started.
0x0002	Ok - Service stopped	Service has been successfully terminated.
0x0010	Response pending	Response pending for another 5 s.
0x0101	Warning - Service running	Service Id used in a *StartService* message refers to an already running service.
0x0102	Warning - Service not running	Service Id used in a *StopService* or *ServicePayload* message refers to a not-running service.
0x0201	Error - Service Launch failed	Service cannot be launched. CDB Source endpoint shall delist the service and send an updated *ServicesSupported* message.
0x0202	Error - Service Termination failed	Service cannot be terminated. CDB Sink endpoint shall ignore *ServicePayload* messages from the service in question. CDB Sink endpoint shall terminate the CDB.
0x0203	Error - Service reset	Service is reset from CDB Source endpoint. Service can be immediately started again using *StartService* message.
0x0204	Error - Service terminated	Service is terminated from CDB Source endpoint. Service is (temporarily) unavailable. CDB Source endpoint shall send an updated *ServicesSupported* message.
0x0205	Error - Unknown Service Id	Service Id used in a *ServicePayload*, *StartService* or *StopService* message does not exist.
0x0206	Error - Wrong Access Control	Access Control in *ServicePayload* is wrong. CDB sink endpoint shall stop the service.
0x0207	Error - Resource busy	Resource constraint service not started. Service resources are busy.
0x0208	Error - Wrong Sequence Number	Sequence number in *PingResponse* is wrong. Receiving CDB endpoint shall terminate CDB.
0x0209	Error - Wrong CDB version number	Wrong version number in the *ServicesRequested* or *ServicesReported* message. CDB shall be terminated.

Response Value		Description
0x020A	Error - Wrong Service version number	Invalid version number in the *StartService* message. Service is not started.
0x020B	Error - Unsupported Payload Format	The payload format used in the *ServicePayload* message is not supported from the Data Service. CDB sink endpoint shall stop the service.

5.8 ByeBye

This message shall be used from either the CDB Source or Sink endpoint to terminate the Common Data Bus endpoint. Table 9 specifies the format of this message:

Table 9: *ByeBye* Message

# bytes	Type	Value	Description
2	U16	0xB109	Message-type
2	U16	0	Payload length

Sending and receiving a *ByeBye* message shall terminate the Common Data Bus, according clause 4.2.1.2.

5.9 Ping

The *Ping* message may be used from either side, to check if the other side is still alive. The endpoint, receiving a ping message, shall reply with a *PingResponse* message containing the sequence number. Table 10 specifies the format of the `Ping` message.

Table 10: *Ping* Message

# bytes	Type	Value	Description
2	U16	0xB108	Message-type
2	U16	2	Payload length
2	U16		Sequence number Arbitrary value for distinguishing *PingResponse* messages.

If a Ping message is not replied via a *PingResponse* message within 10 seconds, the Common Data bus shall be considered non-functional. If either side recognizes the Common Data Bus to be non-functional, it shall terminate the Common Data Bus as specified in clause 4.2.1.2.

5.10 PingResponse

The *PingResponse* message shall be used from either side, to respond to a `Ping` messages. The receiving endpoint should send the *PingResponse* message, prior sending any other message. Table 11 specifies the format of the *PingResponse* message.

Table 11: *PingResponse* Message

# bytes	Type	Value	Description
2	U16	0xB106	Message-type
2	U16	2	Payload length
2	U16		Sequence number in reply to a *Ping* message

6 Message Flow

Table 12 summarizes the CDB Sink and Source Endpoint's message flow, it shall follow. The sending endpoint is given in brackets in the Message Received column.

Table 12: CDB Message Flow

Message Received (from)	Message Responses	Comment
ServicesRequest (Sink)	ServicesSupported	-
	ServiceResponse (0x0209)	Wrong CDB version
ServicesSupported (Source)	No response	Wait for StartService
	ServiceResponse (0x0209)	Wrong CDB version
Start Service (Sink)	ServiceResponse (0x0001)	Launch ok
	ServiceResponse (0x0010)	Response pending
	ServiceResponse (0x0101)	Launch failed - Already running
	ServiceResponse (0x0201) ServicesSupported	Launch failed - Cannot launch service; update service list
	ServiceResponse (0x0205)	Launch failed - Unknown ServiceId
	ServiceResponse (0x0207)	Launch failed - resource busy
	ServiceResponse (0x020A)	Launch failed - Invalid version
Stop Service (Sink)	ServiceResponse (0x0002)	Stop ok
	ServiceResponse (0x0102)	Stop failed - Not running
	ServiceResponse (0x0202)	Stop failed - Cannot stop service
	ServiceResponse (0x0205)	Launch failed - Unknown ServiceId
ServicePayload (Sink)	No response	Payload ok
	ServiceResponse (0x0102)	Payload failed - Service not running
	ServiceResponse (0x0203)	Payload failed - Service reset
	ServiceResponse (0x0204) ServicesSupported	Payload failed - Service terminated; update service list
	ServiceResponse (0x0205)	Payload failed - Unknown Service Id
	ServiceResponse (0x0206)	Payload failed - Wrong Access Control
	ServiceResponse (0x020B)	Payload failed - Unsupported format
ServicePayload (Source)	No response	Payload ok
	ServiceResponse (0x0102)	Payload failed - Service not running
	ServiceResponse (0x0205)	Payload failed - Unknown Service Id
	ServiceResponse (0x0206) StopService	Payload failed - Wrong Access Control; stop service
	ServiceResponse (0x020B) StopService	Payload failed - Unsupported format; stop service
ServiceResponse (Sink)	No action	Other response values
	ByeBye	If response value = 0x0208
	ByeBye	If response value = 0x0209
ServiceResponse (Source)	No action	Other response values
	ByeBye	If response value = 0x0202
	StartService or No action	If response value = 0x0203
	StopService	If response value = 0x0206
	ByeBye	If response value = 0x0208
	ByeBye	If response value = 0x0209
	StopService	If response value = 0x020B

Message Received (from)	Message Responses	Comment
ByeBye (Server Endpoint)	No response	Terminate the CDB session
ByeBye (Client Endpoint)	ByeBye	Terminate the CDB session
Ping (Sink / Source)	PingResponse	-
PingResponse (Sink / Source)	No response	Ping ok
	ServiceResponse (0x0208)	Ping failed - Wrong sequence number

Annex A (informative): Authors and Contributors

The following people have contributed to the present document:

Rapporteur:	Dr. Jörg Brakensiek, E-Qualus (for Car Connectivity Consortium LLC)
Other contributors:	Raja Bose, Nokia Corporation
	Keun-Young Park, Nokia Corporation

History

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