ETSI TS 103 544-6 V1.3.1 (2019-10)

Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 6: Service Binary Protocol (SBP)

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-98-6

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 6 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 Service Binary Protocol (SBP) is a simple, low-bandwidth data service framework, enabling a CDB data service provider and subscriber to utilize common functions like reading, writing or subscribing to objects of a data service.

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] Void.

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

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

[4] ETSI TS 103 544-4 (V1.3.1): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 4: Device Attestation Protocol (DAP)".

[5] ETSI TS 103 544-15 (V1.3.1): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 15: Application Programming Interface (API) Level 1 & 2".

[6] IEEE Std 754-2019™: IEEE Standard for Binary Floating-Point Arithmetic, 22 July 2019.

NOTE: Available at https://standards.ieee.org/standard/754-2019.html.

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
BOM	Byte Order Mark
CDB	Common Data Bus
EBNF	Extended Backus-Naur Form
INT	INTeger
IP	Internet Protocol
LSB	Least Significant Bit
MSB	Most Significant Bit
SBP	Service Binary Protocol
TCP	Transmission Control Protocol
UID	Unique IDentifier
UPnP	Universal Plug-and-Play
USB	Universal Serial Bus
UTC	Coordinated Universal Time
WLAN	Wireless Local Area Network
XML	eXtensible Markup Language

4 MirrorLink® Data Service Architecture

4.1 Overall Architecture

MirrorLink Data service is composed of CDB, service framework and service provider/subscriber. CDB is the underlying multiplexing layer which also provides service discovery feature. On top of it, service framework layer allows implementing a new service in easier way by providing common abstraction for service provider/subscriber.

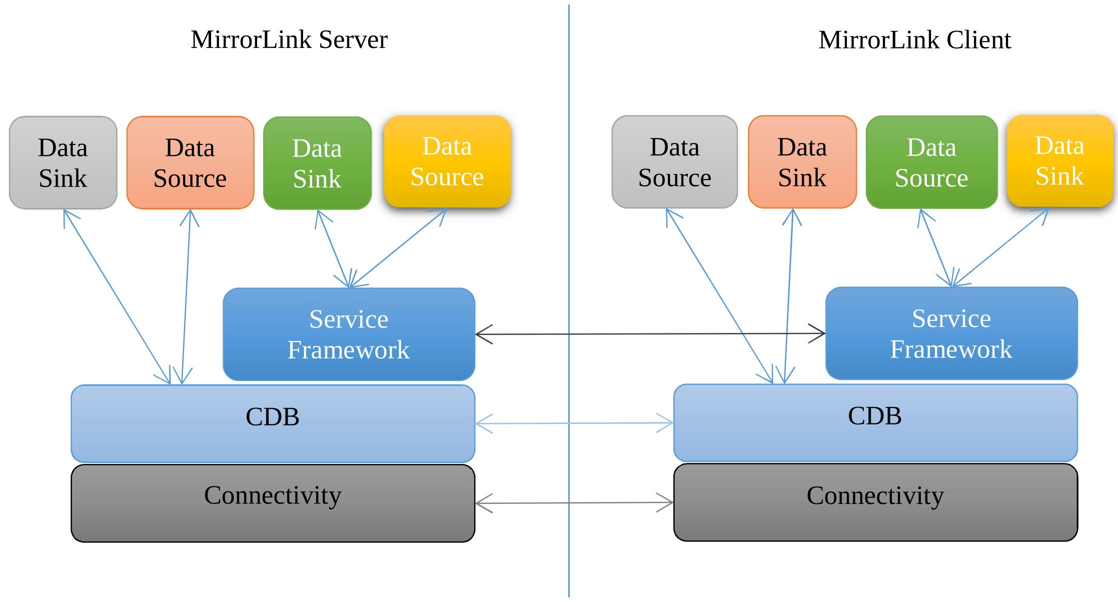

Figure 1: Top level architecture of MirrorLink data service

Figure 1 shows the top level architecture of MirrorLink data service. Underlying connectivity can be a TCP/IP session on top of physical connectivity like USB, WLAN, and Bluetooth. Besides TCP/IP, it will be also possible to run MirrorLink data service 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.

On top of the connectivity layer, the CDB layer is located. CDB relies on the connectivity layer to provide TCP like connection oriented session, and all other layers above rely on the CDB to provide communication interface.

Above the CDB can be the service framework layer or data source (service provider)/data sink (service subscriber) layer depending on the data service used. Service framework layer implements common features for individual data services to allow creating a new service easier. Some data service may decide not to use the service framework to re-use existing protocols or to reduce the additional overhead caused by the framework. It is highly recommended for any new data service to consider using the service framework first. If that approach does not work, accessing directly to CDB layer can be considered. Some data service may open its own TCP/IP session, but such use case is outside the scope of the present document.

On top of data service framework can be service provider (data source) or service subscriber (data sink). Each data source can support up to one data sink. Zero data sink means the service is not used. As there can be only one data sink for each data source, it is up to each side of MirrorLink connection (MirrorLink Server and MirrorLink Client) to make sure that the service can be shared across multiple applications if necessary. Depending on the implementation, there can be one system component which can work as a data sink and can provide received data to all interested applications. Another implementation may allow only one application to get the data. How such access control is implemented is outside the scope of the present document, but it is recommended to allow multiple applications to access the data unless that data is meaningful only for selected applications.

4.2 Version convention

CDB and service framework layers are bound under the same version number provided from CDB. In other words, the service framework layer does not have separate version number. CDB version number will be updated when there is an update in CDB or service framework layer. This present document, goes together with the CDB version 1.1 specification, defined in [2].

All version numbers in MirrorLink data service are composed of major version number and minor version number. A change in the major version number means incompatibility with the previous major version. A change in the minor version guarantees compatibility with the previous minor version. This policy should be maintained across all the layers of MirrorLink data service.

4.3 Starting Data Service

MirrorLink data service requires CDB as underlying layer. And to use data service, CDB should be launched by via UPnP application launch mechanism [3]. Note that there is no separate application for data service, and launching CDB is enough. More details on discovering CDB services can be found from clause 5.3 of CDB specification [2]. Note that MirrorLink Client needs to launch the CDB with right version number.

Once CDB is started, all available services can be discovered by using CDB *ServicesRequest* and *ServicesSupported* messages. Then service client, either from MirrorLink Client or Server side, can ask service server to start the service via CDB *StartService* message. For details, check the CDB specification.

4.4 Data Service Security with Device Attestation Protocol

CDB can support payload encryption by using a pre-arranged session key. In the current MirrorLink architecture, the session key can be acquired after attestation of CDB in MirrorLink Server side by utilizing Device Attestation Protocol [4]. The application public key generated from the attestation of CDB is the session key used for encrypting CDB payload in MirrorLink Client side. MirrorLink Server will use matching private key to decrypt CDB payload. Note that this key can be generated per each MirrorLink connection, and MirrorLink Client shall not re-use the key from the previous connections.

Future versions of the Common Data Bus may provide, additional mechanisms to exchange session keys for encrypting both directions, e.g. symmetric keys.

5 Service Framework: Service BINARY Protocol (SBP)

5.1 Introduction

As a basic data representation mechanism in the service framework layer, we have preferred binary version compared to XML mainly for performance reason. Due to that, a new binary protocol for service framework, SBP (Service Binary Protocol) was defined. Even if the service framework is based on binary protocol, it is important to allow easy service definition and future extendibility. To allow future extension, the concept of identifying each member variable by unique ID is used.

Big-endian is used for all data types. The protocol does not guarantee data alignment for compact data representation, and in most cases, data should be re-constructed from byte stream. Due to that, there is no big advantage of having little-endian instead of big-endian.

SBP assumes lossless data delivery through CDB layer. Due to that, there is no separate data integrity check, but still there can be mal-formed SBP payloads due to implementation error. Such error will be checked inside SBP.

Due to the time constraint for MirrorLink 1.1 specification, decision was made to focus on basic features in this version of specification. Following features will be addressed in this version:

- Getting and setting data.
- Subscribing to a data.

Following features will be added in later revisions:

- Remote Procedure Call feature.
- Details about authentication. Command is defined, but specific details will be added later.
- Meta-data description.
- Interface Description Language: In this draft, style similar to C++ is used for convenience, but it is not formally defined.

5.2 Service Description Example

Service description can be done by defining data objects including member variables. Mechanism for subscribing the data objects will be explained later. Let's assume an example service with the name of `"com.mirrorlink.sensor_example"`. The name is used to uniquely identify the service in CDB layer.

Figure 2 shows data objects defined in the service.

```
/* com.mirrorlink.sensor_example, version 1.0 */

/** @UID: 0xD6804B4A @max_subscription_rate: 50Hz */

Object accelerometer {

    STRUCTURE accel_data {
        FLOAT x;   /// @unit: m/s^2 @mandatory @UID: 0x150A2CB3
        FLOAT y;   /// @unit: m/s^2 @mandatory @UID: 0x150A2CB4
        TIME time; /// @mandatory @UID: 0x00A0FDB2
    };

    STRUCTURE_ARRAY<accel_data> data; /// @UID: 0x144A776F
};

/** @UID: 0xD73DFF88 @writable @control: accelerometer */

Object accelerometer_control {

    BOOLEAN filterEnabled; /// @UID: 0x2B230C64 @optional: false

    INT samplingRate;      /// @UID: 5F2BF0EC
};

/** @UID: 0x41F75401 @max_subscription_rate: 1Hz

Object thermometer {

    INT temperature; /// @UID: 0x9D28234F @unit: Celsius
};
```


Figure 2: Example Service Description

A service can be composed of one or more Objects. The example service is composed of three Objects: *accelerometer*, *accelerometer_control* and *thermometer*. Javadoc style is used to document each object. Each object can be individually accessed by using *Get*, *Set* or *Subscribe* command. Details of these commands will be presented in later clauses.

The *accelerometer* object has one member variable: *data*. The "*data*" is an array of STRUCTURE *accel_data* which has three members: acceleration in x direction, acceleration in y direction, and time. Note that STRUCTURE_ARRAY<XYZ> means an array of STRUCTURE XYZ. Similarly, ARRAY<XYZ> represents an array of basic type (non-STRUCTURE, non-ARRAY) XYZ. The example subscription also shows that the accelerometer object allows the maximum subscription rate of 50 Hz with maximum sampling rate of 100Hz. Due to the difference in rates, one data notification can include multiple samples. Note that /** */ and /// is used for comments and additional information as in Javadoc. All objects allow reading data, but writing is allowed selectively. The *accelerometer_control* object allow writing as *@writable* tag shows. Member variables can be either mandatory or optional. Member variables are mandatory by default, and optional member can be specified with *@optional* tag which can also include the specification of default value when the member variable is not present. For example, in the *accelerometer_control* object, *filterEnabled* is optional with default value of `false`.

Note that *accelerometer_control* object can be used to control the behaviour of accelerometer object.

An Object can inherit other Objects or STRUCTUREs. Then all member variables defined in parents Objects/STRUCTURREs are available in the child Object. A STRUCTURE can also inherit other Objects or STRUCTUREs to re-use the already defined data layout. An example, an Object "A" inheriting a STRUCTURE "a" can be expressed as "Object A inherits STRUCTURE a {};".

5.3 Data representation

Data in SBP is represented as in Table 1 using Extended Backus-Naur Form (EBNF).

Table 1: Binary representation of data in EBNF

EBNF	Form No	Matching *data_type*
data = data_type, value |	1	BOOLEAN, BYTE, SHORT, INT, LONG, FLOAT, DOUBLE
data_type, no_elements, {value} |	2	BYTES, STRING
data_type, element_data_type, no_elements, {value} |	3	ARRAY
data_type, no_elements, { data_with_UID }, END |	4	STRUCTURE
data_type, no_elements, {data}, END;	5	STRUCTURE_ARRAY
data_with_UID = UID, data;	-	-

Each data within a STRUCTURE_ARRAY shall have the same STRUCTURE type, and thus only data with form 4 can be placed.

Table 2 describes symbols used in EBNF description of data.

Table 2: Description of symbols

Category	Size	Description
data_type	U8	Tell the type of data.
UID	U32	Unique identifier of data. Hash value of data's name is used as UID.
value	8, 16, 32, 64 bits	Raw data without any addition. Size depends on the *data_type*.
no_elements	U32	Number of elements contained in the array, array of structure, or structure.
element_data_type	U8	*data_type* of elements contained in the array. This *data_type* can be only BOOLEAN, SHORT, INT, LONG, FLOAT, or DOUBLE. Putting other *data_type* shall be treated as irrecoverable error.
END	U8	Special character (0x81) used for terminating STRUCTURE or STRUCTURE_ARRAY for checking data integrity.
data_with_UID	-	UID, data pair binding UID with data.

Table 3 shows all the data types with matching EBNF description to represent the data.

Table 3: List of *data_type*

Name	data_type	Form	Description
BOOLEAN	0x82	1	U8, true (non-zero) or false (0).
BYTE	0x83	1	8-bit, signed integer.
SHORT	0x84	1	16-bit, signed integer.
INT	0x85	1	32-bit signed integer.