ETSI TS 103 544-22 V1.3.1 (2019-10)

Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 22: Android Specific Specifications enabling AIDL-based MirrorLink® Applications

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-98-22

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 22 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 present document provides the elements of the MirrorLink specification that apply only to Android MirrorLink Server devices.

The API javadoc files contained in the archive CCC-TS-065_Mirrorlink_API-Level2-AIDL-files__v138.zip, contained in ts_10354422v010301p0.zip, are an integral part of the present document.

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 103 544-14 (V1.3.1): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 14: Application Certificates".

[2] 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".

[3] ETSI TS 103 544-16 (V1.3.1): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 16: Application Developer Certificates".

[4] IETF RFC 4648: "The Base16, Base32, and Base64 Data Encodings", October 2006.

NOTE: Available at http://www.ietf.org/rfc/rfc4648.txt.

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

[6] 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".

2.2 Informative references

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

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

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

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

[i.2] Android <manifest> package documentation.

NOTE: Available at http://developer.android.com/guide/topics/manifest/manifest-element.html#package.

[i.3] Android <manifest> android:versionCode documentation.

NOTE: Available at http://developer.android.com/guide/topics/manifest/manifest-element.html#vcode.

[i.4] JAR File Specification.

NOTE: Available at http://docs.oracle.com/javase/7/docs/technotes/guides/jar/jar.html.

[i.5] Signing Your Applications.

NOTE: Available at http://developer.android.com/tools/publishing/app-signing.html.

[i.6] Car Connectivity Consortium: "Android Application ID Generator".

[i.7] Common Intents.

NOTE: Available at https://developer.android.com/guide/components/intents-common.html.

[i.8] Managing audio focus.

NOTE: Available at https://developer.android.com/guide/topics/media-apps/audio-focus.html.

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:

ACMS	Application Certification Management System
AIDL	Android Interface Definition Language
API	Application Programming Interface
APK	Android PacKage
CDB	Common Data Bus

CDMA	Code-Division Multiple Access
HTTP	HyperText Transfer Protocol
IMEI	International Mobile Equipment Identity
IPC	Inter-Process Communication
JAR	Java ARchive
ML	MirrorLink
OCSP	Online Certificate Status Protocol
OS	Operating System
ROM	Read-Only Memory
RSA	Rivest–Shamir–Adleman
SBP	Service Binary Protocol
SDK	Software Development Kit
UID	Unique IDentifier
UPnP	Universal Plug and Play
URL	Uniform Resource Locator
UUID	Universally Unique IDentifier
VNC	Virtual Network Computing

4 Platform-Specific Specification Concept Overview

In order to support third-party applications within a MirrorLink session, the MirrorLink protocols require certain information be provided to the MirrorLink Server's software (the MirrorLink "Stack") and to the Application Certification Management System (ACMS), and that certain functionality be exposed to those applications (the MirrorLink API). In order to prevent fragmentation of the application ecosystem, simplify implementation for MirrorLink Server Device developers, and to increase the number of devices that a given application can be run on, these systems should be common to a given mobile device platform. The goal being that a MirrorLink application written for Android, for example, should be able to run on all MirrorLink-certified Android devices. It is understood that differences of versions and hardware capabilities limit the ability to ensure cross-device compatibility however the intent is that MirrorLink should not create additional barriers to such cross-compatibility.

The present document contains the requirements for MirrorLink Server devices that utilize the Android OS. MirrorLink Server devices that use the Android Operating System shall comply with the requirements listed in the present document.

5 Application Identifier

5.1 General

As described in the Application Certificate Handling specification [1], each application shall have a unique application identifier (App ID) that is provided to the Application Certification Management System (ACMS) via the HTTP GET and OCSP requests sent to it by the MirrorLink Server device. This *AppID* needs to be unique for that application, and change whenever the application is modified. For Android devices, the App ID is generated using the below method. Source code that implements the below algorithm is provided in Annex A.

5.2 Format

The application ID for an Android application shall be a URL-safe base-64 encoding [4] of a SHA-256 digest. As the digest of SHA-256 is 32 bytes long the application ID will therefore be a string of 43 URL-safe base-64 characters.

The application ID shall not place any padding characters at the beginning or end of the encoding. This ensures all implementations will generate an identical application ID and prevents the need to escape any characters when querying the Application Certificate Management System (ACMS).

5.3 Calculation

5.3.1 General

The data to be hashed shall be the concatenation of the following in the order specified:

1) String encoding of Android package name provided in AndroidManifest.xml [i.2].
2) Big endian 8-byte integer representing the version code provided in AndroidManifest.xml [i.3].
3) The string "startManifestMain".
4) For each attribute in the main section of the APK manifest (META-INF/MANIFEST.MF), sorted lexicographically by unicode code point of the attribute name:
   a) String encoding of the full attribute name.
   b) String encoding of the attribute value.
5) The string "endManifestMain".
6) For each file named in the main section of the APK manifest (META-INF/MANIFEST.MF), sorted lexicographically by unicode code point of the file path:
   a) The string "startFile".
   b) Skip this file if the path is "assets/self-signed.ccc.crt".
   c) String encoding of the file path.
   d) For each attribute in the file section of the APK manifest, sorted lexicographically by unicode code point of the attribute name:
      i) String encoding of the full attribute name.
      ii) String encoding of the attribute value.
   e) The string "endFile".

The encoding of strings shall be a big-endian 8-byte integer specifying the length in bytes followed by the UTF-8 encoding of the string.

Example: The string "Hi Σ" would be encoded as the following bytes:

```
0x00 0x00 0x00 0x05          (Length of UTF-8 string, 5 bytes)
0x48 0x69 0x20 0xCE 0xA3     (UTF-8 encoding of string)
```

Therefore, an example APK for an application with the following properties:

- Package name of "com.example.a".
- Version code of 124.
- Containing the following files:
  - assets/image.png.
  - assets/self-signed.ccc.crt.
  - META-INF/CERT.RSA.
  - META-INF/CERT.SF.
  - META-INF/MANIFEST.MF.
  - AndroidManifest.xml.

- classes.dex.
- resources.arsc.

- A META-INF/MANIFEST.MF containing the following:

```
Manifest-Version: 1.0
Created-By: 1.0 (Android)

Name: classes.dex
SHA1-Digest: eEcd5Q6I3GDCkZ7gAAsC0dB8KxU=

Name: resources.arsc
SHA1-Digest: IEBCJEW4Ws8uS/ML7RgEqwGYvtU=

Name: assets/image.png
SHA1-Digest: gHoXviEAT+JdNMqsvo/vERe231Y=

Name: assets/self-signed.ccc.crt
SHA1-Digest: r/HlpR8FRHOKcsF4o5PFSeV32yk=

Name: AndroidManifest.xml
SHA1-Digest: /czwhhK4YJmcwq9E/qHoB26/K90=
```

Would have an application ID which is the SHA-256 digest of the concatenation of the following data where the strings are encoded using the method described above:

```
"com.example.a"
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x7C
"startManifestMain"
"Created-By"
    "1.0 (Android)"
"Manifest-Version"
    "1.0"
"endManifestMain"
"startFile"
"AndroidManifest.xml"
    "SHA1-Digest"
        "/czwhhK4YJmcwq9E/qHoB26/K90="
"endFile"
"startFile"
"assets/image.png"
    "SHA1-Digest"
        "gHoXviEAT+JdNMqsvo/vERe231Y="
"endFile"
"startFile"
"classes.dex"
    "SHA1-Digest"
        "eEcd5Q6I3GDCkZ7gAAsC0dB8KxU="
"endFile"
"startFile"
"resources.arsc"
    "SHA1-Digest"
        "IEBCJEW4Ws8uS/ML7RgEqwGYvtU="
"endFile"
```


The quotation marks are not included in the calculation, they are here just to indicate the exact string which is going to be hashed.

To see a working example for calculating the *AppID* please check the Android Application ID Generator [i.6], which contains a sample APK. The README file provided contains the expected application ID.

The Mirror Link Server should reject any APK which contains the same filename twice. An APK with the same filename multiple times could be used to mask some unwanted data, therefore the Server shall treat it as untrusted.

5.3.2 Shared UIDs

In Android APKs are allowed to share UIDs. Each APK shall be treated as an application as a whole and if two or more APK use the same shared UID, each of them will be considered an application in its own right. Each of those APKs shall be certified independently and each of them will have a unique application ID.

When an application connects to a service of the MirrorLink Server it shall provide the package name of the APK it lives in. The server then shall validate that the package name has the UID of the process making the request. This is because when an IPC request is being made, in Android, there is no way to tell the package from which the request came, only the UID can be retrieved.

5.3.3 ROM applications

Application identifiers of apps built-into the device firmware (aka ROM) may be extracted by the MirrorLink Server from the self-signed certificate instead of being dynamically generated at runtime. This may be necessary due to firmware optimizations, which change the structure of an APK thus making it difficult to maintain a unique application identifier for the same application across different firmware variations. The application identifier of the optimized application within the self-signed certificate shall match the application identifier of the same, un-optimized application.

The application shall contain a self-signed certificate, following the requirements in clause 6.2. The MirrorLink Server shall have an integrity check method for build-in ROM applications to verify the integrity of such build-in applications, before granting the exemption. Integrity check may be done by validating that the ROM application was signed with the same ROM signature key (platform private key).

If an updated version of a ROM-based application is installed as a standard APK, the MirrorLink Server shall treat it as an updated version of the ROM application, shall cease making OCSP requests in relation to the ROM-based version, and shall not list the ROM-based version in the UPnP Application Listings. Updated versions of built-in applications installed as a standard APK shall be treated like normal MirrorLink applications.

5.4 APK Validation

An APK which contains files outside of the 'META-INF' directory with no MANIFEST.MF entries will be rejected by the Android platform. If a file is present in the MANIFEST.MF but does not a have a *-Digest entry then this will also be rejected by the Android platform. This is because both of these are requirements for a valid JAR signature [i.4].

An application ID shall not be generated by an ID generation tool, for an APK with files outside of the 'META-INF' directory with no MANIFEST.MF entry. An application ID also shall not be generated for an APK containing files without a '*-Digest' entry in the MANIFEST.MF.

An application ID shall not be generated also when the same filename is present more than once in the APK. This is to avoid any attempts to hide files within the APK. The MirrorLink Server is responsible itself for rejecting such applications.

A MirrorLink server may validate the signature of the APK in addition to the APK signature being validated by the Android platform.

6 Application Information

6.1 General

As described in clause 6.1 of the Handling of Application Certificates specification [1], an application should come with a MirrorLink developer self-signed certificate Application Certificate for the purposes of providing the information needed by the MirrorLink Server to list the application in the *A_ARG_TYPE_AppList* as described in Table 4.3 of [5], and to control interaction with the ACMS as described in [1]. This clause describes how the Application Certificate is generated and included with the application for Android MirrorLink Server devices, as well as any alternate methods for providing that information.

6.2 Self-Signed Application Certificates

6.2.1 General

The MirrorLink servers running on an Android platform shall allow for Self-Signed Application Certificates as described in clause 6.1 of [1].