FIDO UAF APDU

FIDO Alliance Proposed Standard 20 October 2020

This version:
https://fidoalliance.org/specs/fido-uaf-v1.2-ps-20201020/fido-uaf-apdu-v1.2-ps-20201020.html
Previous version:
https://fidoalliance.org/specs/fido-uaf-v1.2-id-20180220/fido-uaf-apdu-v1.2-id-20180220.html
Editor:
Naama Bak, Morpho
Contributors:
Virginie Galindo, Gemalto
Rolf Lindemann, Nok Nok Labs, Inc.
Ullrich Martini, Giesecke & Devrient
Chris Edwards, Intercede
Jeff Hodges, Paypal

The English version of this specification is the only normative version. Non-normative translations may also be available.

Abstract

This specification defines a mapping of FIDO UAF Authenticator commands to Application Protocol Data Units (APDUs) thus facilitating UAF authenticators based on Secure Elements.

Status of This Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current FIDO Alliance publications and the latest revision of this technical report can be found in the FIDO Alliance specifications index at https://fidoalliance.org/specifications/.

This document was published by the FIDO Alliance as a Proposed Standard. If you wish to make comments regarding this document, please Contact Us. All comments are welcome.

Implementation of certain elements of this Specification may require licenses under third party intellectual property rights, including without limitation, patent rights. The FIDO Alliance, Inc. and its Members and any other contributors to the Specification are not, and shall not be held, responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights.

THIS FIDO ALLIANCE SPECIFICATION IS PROVIDED “AS IS” AND WITHOUT ANY WARRANTY OF ANY KIND, INCLUDING, WITHOUT LIMITATION, ANY EXPRESS OR IMPLIED WARRANTY OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

This document has been reviewed by FIDO Aliance Members and is endorsed as a Proposed Standard. It is a stable document and may be used as reference material or cited from another document. FIDO Alliance's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment.

Table of Contents

1. Notation

Type names, attribute names and element names are written as `code`.

String literals are enclosed in “”, e.g. “UAF-TLV”.

In formulas we use “|” to denote byte wise concatenation operations.

The notation `base64url(byte[8..64])` reads as 8-64 bytes of data encoded in base64url, "Base 64 Encoding with URL and Filename Safe Alphabet" [RFC4648] *without padding*.

UAF specific terminology used in this document is defined in [FIDOGlossary].

All diagrams, examples, notes in this specification are non-normative.

All TLV structures defined in this document MUST be encoded in little-endian format.

All APDU defined in this document MUST be encoded as defined in [ISOIEC-7816-4-2013].

1.1 Key Words

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

2. Introduction

This section is non-normative.

This specification defines the interface between the FIDO UAF Authenticator Specific Module (ASM) [UAFASM] and authenticators based upon "Secure Element" technology. The applicable secure element form factors are UICC (SIM card), embedded Secure Element (eSE), µSD, NFC card, and USB token. Their common characteristic is they communicate using Application Programming Data Units (APDU) in compliance with [ISOIEC-7816-4-2013].

Implementation of this specification is optional in the UAF framework, however, products claiming to implement the transport of UAF messages over APDUs should implement it.

This specification first describes the various fashions in which Secure Elements can be incorporated into UAF authenticator implementations — known as *SE-based authenticators* or just *SE authenticators* — and which components are responsible for handling user verification as well as cryptographic operations. The specification then describes the overall architecture of an SE-based authenticator stack from the ASM down to the secure element, the role of the "UAF Applet" running in the secure element, and outlines the nominal communication flow between the ASM and the SE. It then defines the mapping of UAF Authenticator commands to APDUs, as well as the FIDO-specific variants of the VERIFY APDU command.

> NOTE
>
> This specification does not define how an SE-based authenticator stack may be implemented, e.g., its integration with TEE or biometric sensors. However, SE-based authenticator vendors should reflect such implementation characteristics in the authenticator metadata such that FIDO Relying Parties wishing to be informed of said characteristics may have access to it.

3. SE-based Authenticator Implementation Use Cases

This section is non-normative.

Secure elements can be leveraged in different scenarios in the UAF technology. It can support user gestures (used to unlock access to FIDO credentials) or it can be involved in the actual cryptographic operations related to FIDO authentication. In this specification, we will be considering the following SE-based authenticator implementation use cases:

1. The Secure Element (SE) *is* the (silent) Authenticator.
2. The SE is part of the Authenticator which is composed of a Trusted Application (TEE) based User Verification component, potentially a TEE based transaction confirmation display and the crypto kernel inside the SE (***Hybrid SE Authenticator***).
3. The authenticator (Hybrid SE Authenticator) consists of
    - the SE implementing the matcher and the crypto kernel
    - and a specific software module (e.g. running on the FIDO User Device) to capture the user verification data (e.g. PIN, Face, Fingerprint).

3.1 Hybrid SE Authenticator

In FIDO UAF, the access to credentials for performing the actual authentication can be protected by a user verification step. This user verification step can be based on a PIN, a biometric or other methods. The authenticator functionality might be implemented in different components, including combinations such as TEE and SE, or fingerprint sensor and SE. In that case the SE implements only a part of the authenticator functionality.

> NOTE
>
> The reason for using such hybrid configuration is that Secure Elements do not have any user interface and hence cannot directly distinguish physical user interaction from programmatic communication (e.g. by malware). The ability to require a physical user interaction that cannot be emulated by malware is essential for protecting against scalable attacks (see [FIDOSecRef]). On the other hand, TEEs (or biometric sensors implemented in separate hardware) which can provide a trusted user interface typically do not offer the same level of key protection as Secure Elements.
>
> Strictly spoken, a Hybrid SE Authenticator (voluntarily) uses the Authenticator Command interface [UAFAuthnrCommands] *inside* the authenticator, e.g. between the crypto kernel and the user verification component.

Examples of Hybrid SE authenticators are:

1. User PIN code capture and verification are implemented entirely in a TEE relying on Trusted User Interface and secure storage capabilities of the TEE and, once the PIN code is verified, the FIDO UAF crypto operations are performed in the SE.
2. User fingerprint is captured via a fingerprint sensor, the fingerprint match is performed in the TEE, relying on matching algorithms. Once the fingerprint has been positively checked, the cryptographic operations are executed in the Secure Element.
3. The user verification is implemented as match-on-chip in separate hardware and FIDO UAF cryptographic operations are implemented in the SE.

In all those cases, the hybrid nature of the authenticator will be managed by the software-based host, regardless of its nature (TEE, SW, Biometric sensor..). There are a number of possible interactions between the ASM and the SE actually implementing the verification and the cryptographic operations to consider within those use cases.

1. PIN user verification where the user interaction for the PIN entry is performed externally to the SE. The PIN may then be passed within a VERIFY command to the SE, followed by the actual cryptographic operations (such as the Register and Sign UAF authenticator commands).
2. Biometric user verification where the sample capture and matching is performed externally to the SE (e.g. in TEE or in a match-on-chip FP sensor). This would then only need to send to the SE the actual cryptographic operation needed in this session (such as the Register and Sign UAF authenticator commands).
3. User verification sample (Faceprint, Fingerprint..) capture is performed externally to the SE. The sample is then sent to a match-on-card applet in the SE that behaves as a global PIN to enable access to the cryptographic operation required within this session.

3.1.1 Architecture of the Hybrid SE Authenticator

In order to support an Hybrid SE Authenticator, a dedicated software-based host MUST be created which knows how the SE applet works. The communication between the SE applet and the host is defined based on [ISOIEC-7816-4-2013]. Whether a PC or mobile device the architecture is still the same, as defined below:

- `Application Layer` : This component is responsible for acquiring the user verification sample and mapping UAF commands to APDU commands.
- `Communication layer` : This is the [ISOIEC-7816-4-2013] APDUs interface, which provides methods to list and select readers, connect to a Secure Element and interact with it.
- `SE Access OS APIs` : OMA, PC/SC, NFC API, CCID…
- `Secure Element` : UICC, micro SD, eSE, Dual Interface card…

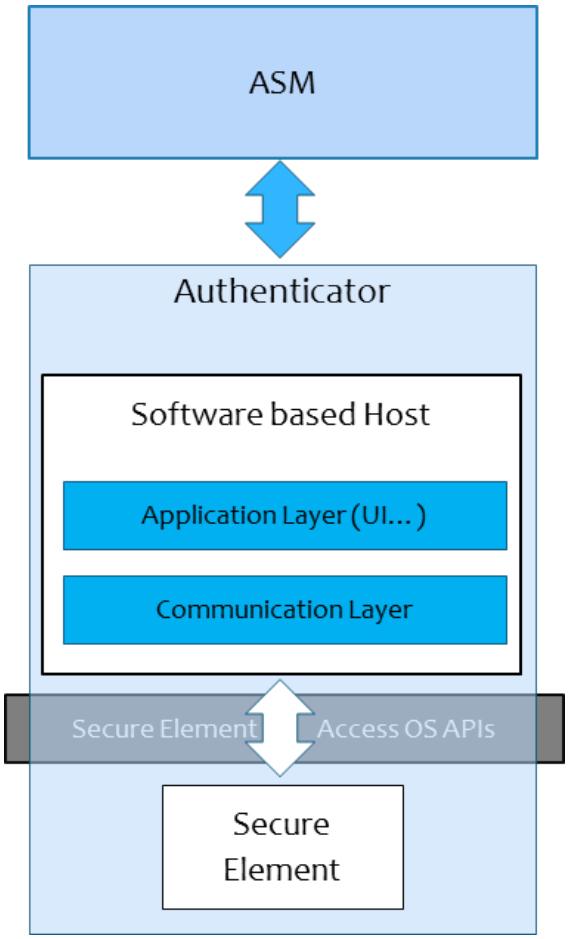

Fig. 1 Architecture of Hybrid SE Authenticator

APDU command-response paire are handled as indicated in [ISOIEC-7816-4-2013].

3.1.2 Communication flow between the ASM and the Hybrid SE Authenticator

The host is the entity communicating with the SE and which knows how the SE and the applet running in the SE can be accessed. The host could be a Trusted Application (TA) which runs inside a TEE or simply an application which runs in the normal world.

The following diagram illustrates how the Host of the Hybrid SE Authenticator MAY map the UAF commands to APDU commands. In this diagram, the User Verification Module is considered inside the SE applet.

NOTE

If the User Verification Module is inside the Host, for example in the context of the TEE, the `UserVerificationToken` shall be generated in the Host and not in the SE. As a result step 6 (Figure 2) should be executed in the Host instead of the SE.

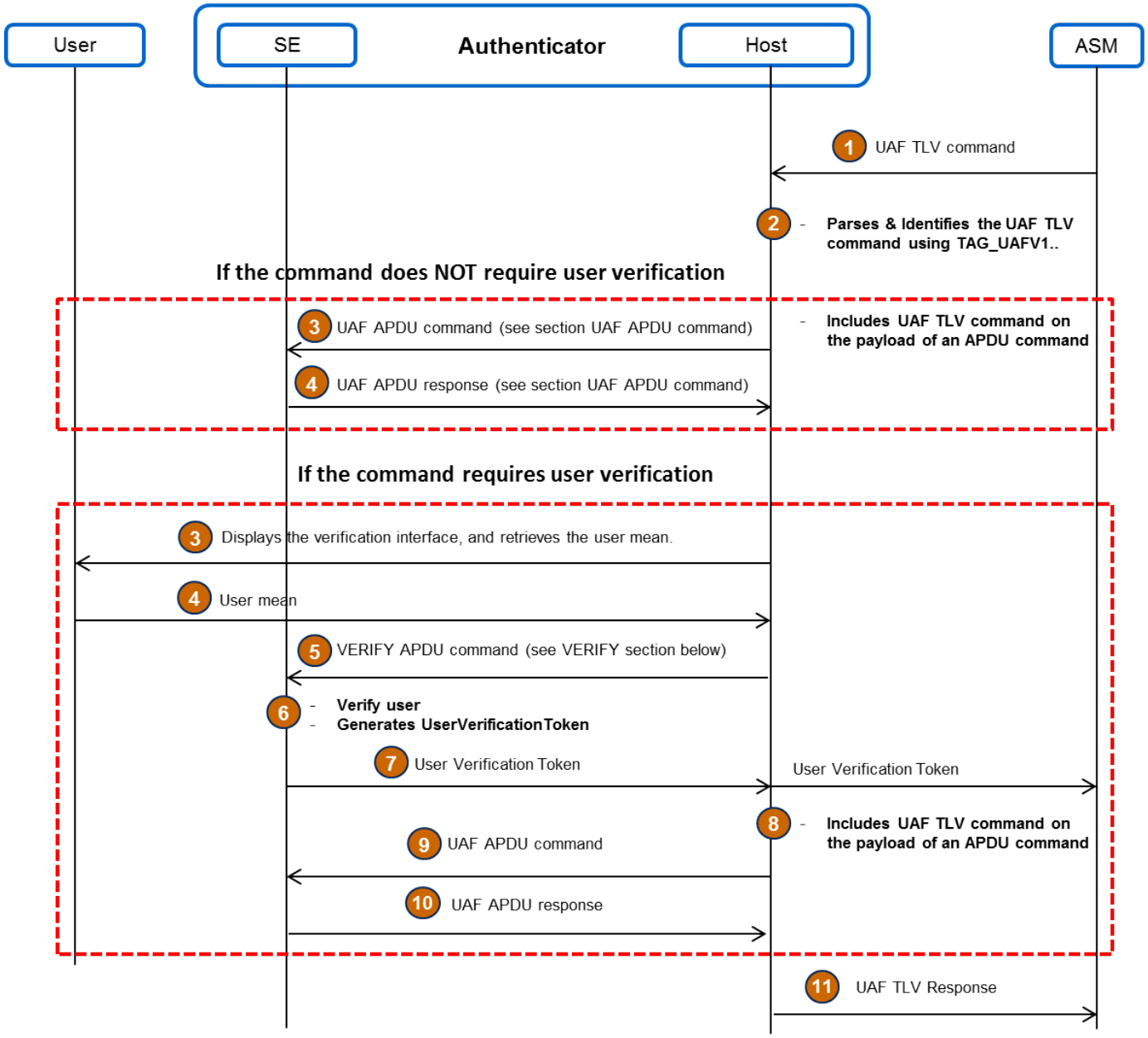

Fig. 2 Communication flow between the ASM and the Hybrid SE Authenticator

4. FIDO UAF Applet and APDU commands

This section is normative.

4.1 UAF Applet in the Authenticator

4.1.1 Application Identifier

The FIDO UAF AID is defined in [UAFRegistry].

4.1.2 User Verification

The User verification is based on the submission of a PIN/password (i.e., knowledge based) or a biometric template (i.e., biometric based).

In this document, the envisaged user verification methods are PIN and biometric based.

4.1.3 Cryptographic operations

The SE applet must be able to perform a set of cryptographic operations, such as key generation and signature computation. The cryptographic operations are defined in [UAFAuthnrCommands]. The SE applet SHALL use the cryptographic algorithms indicated in [UAFRegistry].

4.2 APDU Commands for FIDO UAF

4.2.1 Class byte coding

CLA indicates the class of the command.

Commands	CLA
SELECT, VERIFY (ISO Version), GET RESPONSE (ISO Version)	0x00
VERIFY, UAF, GET RESPONSE	0x80

Table 1: Class byte coding

> NOTE
>
> If the payload of an APDU command is longer than 255 bytes, command chaining as described in [ISOIEC-7816-4-2013] should be used, even though CLA is proprietary.

4.2.2 APDU command "UAF"

4.2.2.1 Mapping between FIDO UAF authenticator commands and APDU commands

This section describes the mapping between FIDO UAF authenticator commands and APDU commands.

The mapping consists of encapsulating the entire UAF Authenticator Command in the payload of the APDU command, and the UAF Authenticator Command response in the payload of the APDU Response.

The host SHALL set the INS byte to **"0x36"** for all UAF commands The SE SHALL read the UAF command number and data from the payload in the data part of the command.

The payload of the APDU command is encoded according to [UAFAuthnrCommands], the first 2 bytes of each command are the UAF command number. Upon command reception, the SE applet MUST parse the first TLV tag (2 bytes) and figure out which UAF command is being issued. The SE applet SHALL parse the rest of the FIDO Authenticator Command payload according to [UAFAuthnrCommands].

The mapping of UAF Authenticator Commands to APDU commands is defined in the following table:

CLA	INS	P1	P2	Lc	Data In	Le
Proprietary(See Table 1)	0x36	0x00	0x00	Variable	UAF Authenticator Command structure	None

Table 2: UAF APDU command

The UAF Authenticator Command structures are defined in part 6.2 of [UAFAuthnrCommands].

> NOTE
>
> If the `UserVerificationToken` is supported, The ASM must set the `TAG_USERVERIFY_TOKEN` flag in the value of the `UserVerificationToken`, received previously contained in either a `Register` or `Sign` command. Please refer to the **FIG 1** in Use-Case section.

4.2.2.2 Response message and status conditions of an "UAF" APDU command

The status word of an "UAF" APDU response is handled at the Host level; the host must interpret and map the status word based on the table below.

If the status word is equals to **"9000"**, the host shall return back to the ASM the entire data field of the APDU response. It the status word is "61xx", the host shall issue `GET RESPONSE` (see below) until no more data is available, concatenate these response parts and then return the entire response. Otherwise, the host has to build an UAF TLV response with the mapped status codes `TAG_STATUS_CODE`, using the following table.

For example, if the status word returned by the Applet is **“6A88”**, the host shall put `UAF_CMD_STATUS_USER_NOT_ENROLLED` in the status codes of the UAF TLV response.

APDU STATUS CODE	FIDO UAF STATUS CODE	NAME	DESCRIPTION
9000	0x00	UAF_CMD_STATUS_OK	Success.
61xx	0x00	UAF_CMD_STATUS_OK	Success, xx bytes available for GET RESPONSE.
6982	0x02	UAF_CMD_STATUS_ACCESS_DENIED	Access to this operation is denied.
6A88	0x03	UAF_CMD_STATUS_USER_NOT_ENROLLED	User is not enrolled with the authenticator.
N/A	0x04	UAF_CMD_STATUS_CANNOT_RENDER_TRANSACTION_CONTENT	Transaction content cannot be rendered.
N/A	0x05	UAF_CMD_STATUS_USER_CANCELLED	User has cancelled the operation.
6400	0x06	UAF_CMD_STATUS_CMD_NOT_SUPPORTED	Command not supported.
6A81	0x07	UAF_CMD_STATUS_ATTESTATION_NOT_SUPPORTED	Required attestation not supported.
6A80	0x08	UAF_CMD_STATUS_PARAMS_INVALID	The request was rejected due to an incorrect data field.
6983	0x09	UAF_CMD_STATUS_KEY_DISAPPEARED_PERMANENTLY	The UAuth key which is relevant for this command disappeared from the authenticator and cannot be restored.
N/A	0x0a	UAF_CMD_STATUS_TIMEOUT	The operation in the authenticator took longer than expected.
N/A	0x0e	UAF_CMD_STATUS_USER_NOT_RESPONSIVE	The user took too long to follow an instruction.
6A84	0x0f	UAF_CMD_STATUS_INSUFFICIENT_RESOURCES	Insufficient resources in the authenticator to perform the requested task.
63C0	0x10	UAF_CMD_STATUS_USER_LOCKOUT	The operation failed because the user is locked out and the authenticator cannot automatically trigger an action to change that.
All other codes	0x01	UAF_CMD_STATUS_ERR_UNKNOWN	An unknown error

Table 3: Mapping between APDU Status Codes and FIDO Status Codes [UAFAuthnrCommands]

The response message of an UAF APDU command is defined in the following table :

Data field	SW1 - SW2
not present	**“6982”** – The request was rejected due to user verification being required. **“6A80”** – The request was rejected due to an incorrect data field. **“6A81”** – Required attestation not supported **“6A88”** – The user is not enrolled with the SE **“6400”** – Execution error, undefined UAF command **“6983”** – Authentication data not usable, Auth key disappeared

UAF Authenticator Command response [UAFAuthnrCommands]	**“61xx”** – Success, xx bytes available for GET RESPONSE. **“9000”** – Success

Table 4: Response message of an "UAF" APDU command

4.2.3 APDU Command "SELECT"

A successful SELECT AID allows the host to know that the applet is active in the SE, and to open a logical channel with this end.

In Android smartphones apps are not allowed to use the basic channel to the SIM because this channel is reserved for the baseband processor and the GSM/UMTS/LTE activities. In this case the app must select the applet in a logical channel.

The host must send a `SELECT APDU` command to the SE applet before any others commands.

As a result, the command for selecting the applet using the FIDO UAF AID is :

CLA	INS	P1	P2	Lc	Data In	Le
0x00	0xA4	0x04	0x0C	0x08	0xA000000647AF0001	No response data is requested if the SELECT command's "Le" field is absent. Otherwise, if the "Le" field is present, vendor-proprietary data is being requested.

Table 5: SELECT AID command

4.2.4 APDU Command "VERIFY"

This command is used to request access rights using a PIN or Biometric sample. The SE applet shall verify the sample data given by the Host against the reference PIN or Biometric held in the SE.

Please refer to [ISOIEC-7816-4-2013] and [ISOIEC-19794] for Personal verification through biometric methods.

If the verification is successful and `UserVerificationToken` is supported by the SE applet, a token SHALL be generated and sent to the Host. Without having this token, the Host cannot invoke special UAF commands such as Register or Sign.

The support of `UserVerificationToken` can be checked by examining the contents of the `GetInfo` response in the `AuthenticatorType` TAG or the response of `SELECT APDU` command [UAFAuthnrCommands].

Refer to [FIDOGlossary] for more information about `UserVerificationToken`.

4.2.4.1 Command structure

CLA	INS	P1	P2	Lc	Data In	Le
ISO or Proprietary: see [ISOIEC-7816-4-2013]	0x20 (for PIN) or 0x21 (for biometry)	0x00	0x00	Variable	Verification data	None or expected Le for `UserVerificationToken`

Table 6: VERIFY command encoding for PIN verification

4.2.4.2 Response message and status conditions

Data Out	SW1 - SW2
Absent (ISO-Variant) or `UserVerificationToken` (proprietary)	See [ISOIEC-7816-4-2013]

Table 7: Response message and status conditions

> NOTE
>
> An SE applet that does not support `UserVerificationToken`, may use the [ISOIEC-7816-4-2013] VERIFY command. In this case, the VERIFY command must be securely bound to `Register` and `Sign` commands, so a secure bound method shall be implemented in the SE applet, such as Secure Messaging.

4.3 Managing Long APDU Commands and Responses

If a Secure Element is able to send a complete response (e.g. extended length APDU, block chaining), `GET RESPONSE` APDU command SHALL be

used, as defined in ISO Variant section. Otherwise, the proprietary solution SHALL be used, as defined in section Proprietary Variant.

4.3.1 ISO Variant

The [ISOIEC-7816-4-2013] GET RESPONSE command is used in order to retrieve big data returned by APDU command "UAF".

4.3.2 Proprietary Variant

In order to avoid using Get Response APDU command which is not supported by all devices and terminals, a propriatry method is defined for managing the long data answers at application level.

When using the proprietary variant, the response to the UAF APDU command SHALL include the Tag **"0x2813"**, that specifies the length of the response.

Response Data Out description

Tag
: **0x2813**

Length
: variable (2 bytes)

Value
: Expected data length (2 bytes)

In the case where the data does not fit into a single Data Out message, the host SHALL repeat the "UAF" command with P2 = 1 value mentioning this is a repetition of the incoming APDU to get all the data. This process SHALL be repeated until the entire data are collected by the host.

Here is an example of an APDU Response which contains more than 255 bytes in the payload.

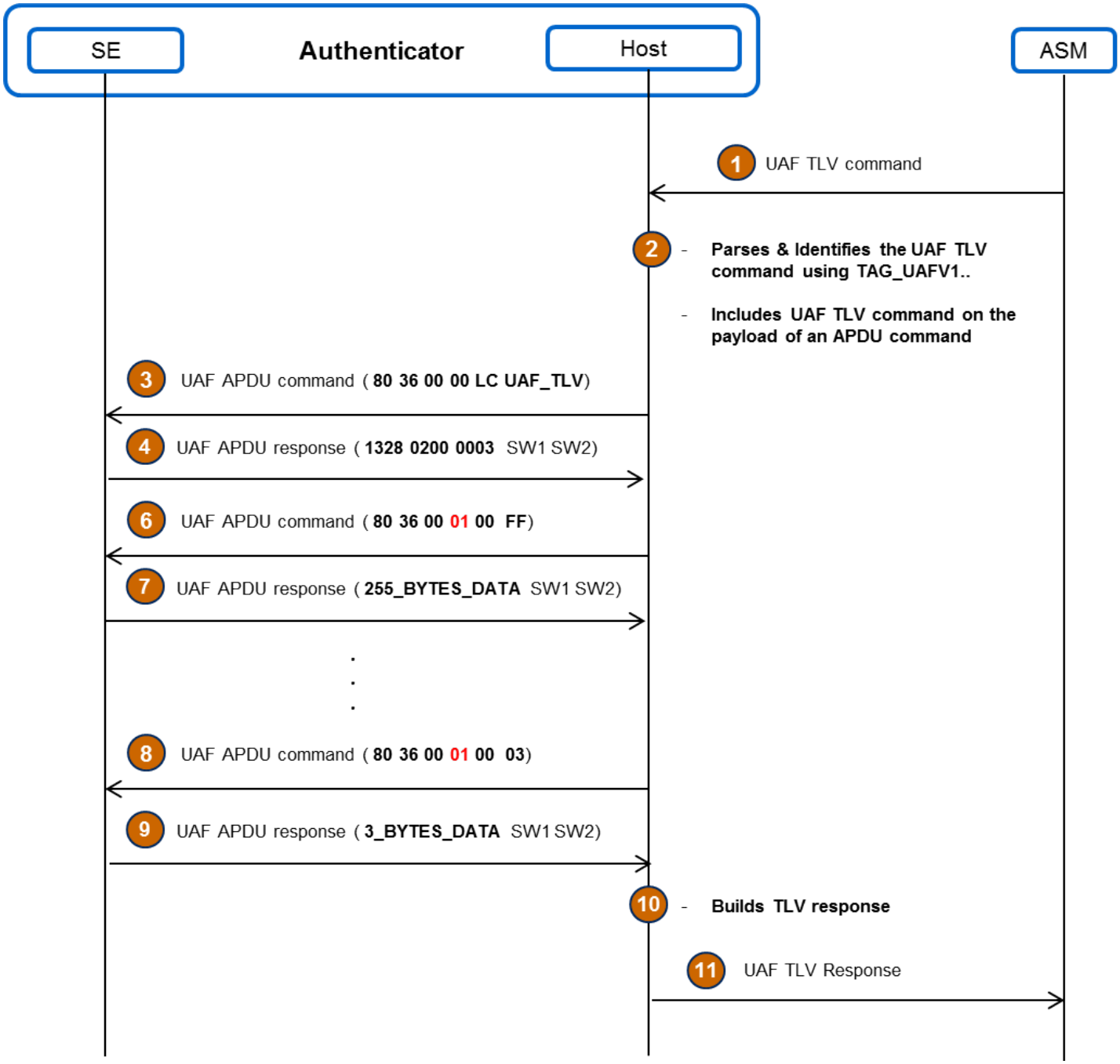

Fig. 3 Long APDU management using the defined proprietary method

> NOTE
>
> The host shall support both versions of Get Response APDU command, and figure out which command must be sent to the Applet by parsing the response of the UAF APDU command. If the UAF APDU command response contains the Tag **"0x2813"**, the host must send a proprietary Get Response APDU command, otherwise the host must send the ISO variant of Get Response APDU command.

5. Security considerations

This section is non-normative.

Guaranteeing trust and security in a fragmented architecture such as the one levering on SE is a challenge that the Host has to address regardless of its nature (TEE or Software based), which results in different challenges from a security and architecture perspective. One could list the following ones:

- use of a trusted user interface to enter a PIN on the device,
- secure transmission of PIN or fingerprint minutiae,

- minutiae extraction format,
- integrity of data transmitted between a Host and a SE.

Hence, we will only consider here, security challenges affecting the interface between the Host and the SE.

A possible way to maintain the integrity and confidentiality when APDUs commands are exchanged is to enable a secure channel between the Host and the SE. While this is left to implementation, there are several technologies allowing to build a secure channel between a SE and a devices, that may be implemented.

- Secure channel between a trusted application in a TEE and an applet in a SE [GlobalPlatform-TEE-SE].
- Secure channel between a device and an applet in a secure element [GlobalPlatform-Card].
- Secure channel between a device and a SE [ETSI-Secure-Channel].

A. References

A.1 Normative references

[RFC4648]
S. Josefsson. *The Base16, Base32, and Base64 Data Encodings (RFC 4648)*. October 2006. URL: http://www.ietf.org/rfc/rfc4648.txt

A.2 Informative references

[ETSI-Secure-Channel]
. *ETSI TS 102 484 Smart Cards; Secure channel between a UICC and an end-point terminal*. URL:

[FIDOGlossary]
R. Lindemann; D. Baghdasaryan; B. Hill; J. Hodges. *FIDO Technical Glossary*. Review Draft. URL: https://fidoalliance.org/specs/fido-v2.0-id-20180227/fido-glossary-v2.0-id-20180227.html

[FIDOSecRef]
R. Lindemann; D. Baghdasaryan; B. Hill; J. Hill; D. Biggs. *FIDO Security Reference*. 27 February 2018. Implementation Draft. URL: https://fidoalliance.org/specs/fido-v2.0-id-20180227/fido-security-ref-v2.0-id-20180227.html

[GlobalPlatform-Card]
. *Secure Channel Protocol 03 – GlobalPlatform Card Specification v.2.2 – Amendment D*. URL:

[GlobalPlatform-TEE-SE]
. *TEE Secure Element API Specification v1.0 | GPD_SPE_024*. URL:

[ISOIEC-19794]
. *ISO 19794: Information technology - Biometric data interchange formats*. URL:

[ISOIEC-7816-4-2013]
. *ISO 7816-4: Identification cards – Integrated circuit cards; Part 4 : Organization, security and commands for interchange*. URL:

[RFC2119]
S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. March 1997. Best Current Practice. URL: https://tools.ietf.org/html/rfc2119

[UAFASM]
D. Baghdasaryan; J. Kemp; R. Lindemann; B. Hill; R. Sasson. *FIDO UAF Authenticator-Specific Module API*. Review Draft. URL: https://fidoalliance.org/specs/fido-uaf-v1.2-ps-20201020/fido-uaf-asm-api-v1.2-ps-20201020.html

[UAFAuthnrCommands]
D. Baghdasaryan; J. Kemp; R. Lindemann; R. Sasson; B. Hill; J. Hodges; K. Yang. *FIDO UAF Authenticator Commands*. Review Draft. URL: https://fidoalliance.org/specs/fido-uaf-v1.2-ps-20201020/fido-uaf-authnr-cmds-v1.2-ps-20201020.html

[UAFRegistry]
R. Lindemann; D. Baghdasaryan; B. Hill. *FIDO UAF Registry of Predefined Values*. Review Draft. URL: https://fidoalliance.org/specs/fido-v2.0-id-20180227/fido-registry-v2.0-id-20180227.html