Federal Office
for Information Security

Developer Documentation FSP - Functional Specification

de.fac2 - FIDO U2F Authenticator Applet, v1.34

1.1 (3. Apr. 2019)

Federal Office for Information Security
Post Box 20 03 63
D-53133 Bonn

Internet: https://www.bsi.bund.de
© Federal Office for Information Security 2020

Table of Contents

Figures

Tables

1 Interface description

The only interfaces this TOE supports is an interface which receives and responses APDU as specified in ISO7816-4. The TOE receives incoming commands in a command APDU which follows the structure defined in ISO/IEC 7816-4. It consists of a four byte header and a conditional body of variable length. A response APDU is sent by the TOE to the reader – it contains from 0 to 65.536 bytes of data and 2 mandatory status bytes (SW1, SW2).

Command APDU		
Field name	**Length (bytes)**	**Description**
CLA	1	Instruction class - indicates the type of command
INS	1	Instruction code - indicates the specific command
P1-P2	2	Instruction parameters for the command
L_c	0, 1 or 3	Encodes the number (N_c) of bytes of command data to follow 0 bytes denotes N_c=0 1 byte with a value from 1 to 255 denotes N_c with the same value 3 bytes, the first of which must be 0, denotes N_c in the range 1 to 65 535 (all three bytes may not be zero)
Command data	N_c	N_c bytes of data
L_e	0, 1, 2 or 3	Encodes the maximum number (N_e) of response bytes expected 0 bytes denotes N_e=0 1 byte in the range 1 to 255 denotes that value of N_e, or 0 denotes N_e=256 2 bytes (if L_c was present in the command) in the range 1 to 65 535 denotes N_e of that value, or two zero bytes denotes 65 536 3 bytes (if L_c was not present in the command), the first of which must be 0, denote N_e in the same way as two-byte L_e
Response APDU		
Response data	N_r	Response data
SW1-SW2	2	Command processing status,

The TOE accepts the following CLA bytes:

- 0x00: Standard CLA byte for all instructions that will be available in the “state ready to use”.
- 0x01: Proprietary CLA byte for setting up the attestation certificate while the card is in state “uninitialized”

The TOE will accept the following instruction bytes

- 0x01: U2F_REGISTER
- 0x02: U2F_AUTHENTICATE
- 0x03: U2F_GET_VERSION
- 0x09: SET_ATTESTATION_CERT
- 0xC0: GET_RESPONSE
- 0x8E: RESET

The TOE is able to process standard APDU length as well as extended length APDUs for all TSFIs. If the reader wants to use extended length APDUs he can use 2 or 3-byte sized length. The TOE will then also answer in extended length response APDU.

In the following TSFI descriptions all length values will be noted in one byte which implies a standard size APDU. This length values can be also code in 2 or 3-byte sized length if the reader wants to use extended length. The coding of the length value doesn't have any impact on the behavior or results of any function other than the sizes that will be transceived in one APDU

Each of the instruction which the TOE provides is one TSFI. The TSFIs will be described in this chapter.

1.1 TSFI U2F_REGISTER

This TSFI is used to initiate a U2F token registration. The registration request has the following structure and parameters:

CLA	INS	P1	P2	L_c	Data	L_e
0x00	0x01	0x00	0x00	0x40	challenge parameter || application parameter	0x00

The parameters P1 and P2 will not be checked and have no further function. They shall both be set to 0x00.

The data field of this APDU has two parts:

- 32 byte challenge parameter and
- 32 byte application parameter

The values of these data parts is specified in the [FIDO U2F Raw] specification.

This TSFI first checks if the data field contains exactly 64 bytes, enforce user presence check and check if the internal FIDO signing counters limit isn't reached. Then it builds a key handle and the users FIDO EC public key and returns it together with the attestation certificate and a signature as specified in FIDO specification [FIDO U2F Raw]. If the registration was successful the SW will be 0x9000 (or 0x61xx, see TSFI GET_RESPONSE) and the data field in the response APDU is a concatenation of the following values :

- 1 byte with value 0x05.
- 65 bytes FIDO user public key
- 1 byte key handle length byte, which specifies the length of the key handle.
- key handle with length specified in previous field.
- attestation certificate with variable length.
- A ECDSA signature [variable length, 71-73 bytes].

More detailed information can be found in the [FIDO U2F Raw] specification.

If an exception occurs the data field in the response APDU will be empty. The SW bytes then will be either are

- SW 0x6700 if data field doesn't contain exactly 64 bytes
- or SW 0x6985 if user presence is required
- or SW 0x6a84 if signing counter exceed its limit

This is a **SFR-enforcing** interface because it performs one of the main function of this applet. It performs the user presence check, uses the platforms signing function and hashes data to perform the U2F Authentication process.

1.2 TSFI U2F_AUTHENTICATE

This TSFI is used to initiate a U2F token authentication. The authentication request has the following structure and parameters:

CLA	INS	P1	P2	L_c	Data	L_e
0x00	0x02	0x07 or 0x03	0x00	Data length	challenge parameter || application parameter || key handle length || key handle	0x00

The parameter P1 controls what actions the TOE will perform. Only value 0x07 or 0x03 are valid values. All other values will lead to SW 0x6A86 and no processing of any input data will be performed. See below or in specification [FIDO U2F Raw] for further details on P1.

The parameter P2 will not be checked and has no further function. It shall be set to 0x00.

The data field of this APDU has five parts:

- 32 byte challenge parameter and
- 32 byte application parameter and
- 1 byte key handle length and
- key handle with length specified in the previous data part

Details of the values of these data parts are specified in [FIDO U2F Raw].

This TSFI first checks if the data field contains at least 65 bytes, enforce user presence check (depends on parameter P1) and check if the internal FIDO signing counters limit isn't reached. Depending on P1 the TOE will perform different actions:

If P1 is 0x07 this function will only check if the given keyHandle contains a valid MAC. If MAC was correct SW 0x6985 will be return in order to signal that keyHandle is valid and matches with the given challenge parameter and application parameter. Otherwise, SW 0x6a80 will be return in order to signal that the keyHandle is unknown or invalid.

If P1 is 0x03 or 0x08 this function will first check if the given keyHandle contains a valid MAC. If the MAC is valid the TOE will derivate the users FIDO EC private key and build a signature over the following data: application parameter, user presence indicator, FIDO signing counter, challenge parameter. The signature will be returned together with the user presence flag and the FIDO signing counter in the data field of the

Response APDU. If the authentication was successful the SW will be 0x9000 (or 0x61xx, see TSFI GET_RESPONSE) and the data field in the response APDU is a concatenation of the following values:

- 1 byte user presence flag
- 4 bytes FIDO signing counter
- Signature of variable length

If an exception occurs the data field in the response APDU will be empty. The SW bytes then will be either are:

- SW 0x6700 if data field contains less than 65 bytes
- SW 0x6a86 if P1 contains invalid data
- SW 0x6a84 if signing counter exceed its limit
- SW 0x6a80 if key handle verification failed
- SW 0x6985 if user presence is required, could also signal success conditions for "check only" operation if P1 was set to 0x07

This is a **SFR-enforcing** interface because it performs one of main function of this applet. It performs the user presence check, uses the platforms signing function and macs data to perform the U2F Authentication process.

1.3 TSFI U2F_GET_VERSION

This TSFI is used to request the U2F protocol version that is implemented. The version request has the following structure and parameters:

CLA	INS	P1	P2	L_c	Data	L_e
0x00	0x03	0x00	0x00	-	-	0x00

The parameters P1 and P2 will not be checked and have no further function. They shall both be set to 0x00.

This APDU expects no data and will ignore incoming data.

The reponse APDU will return the ASCII representation of the version string 'U2F_V2' in the data field and SW 0x9000.

Because this interface doesn't perform any SFR relevant actions, this interface is **SFR non-interfering**.

1.4 TSFI SET_ATTESTATION_CERT

This TSFI is used to set the TOEs Attestation certificate. This function is only available as long as the certificate wasn't transferred completely to the TOE. Once the TOE is in state READY_FOR_USE, the instruction can't be used anymore. The command has the following structure and parameters:

CLA	INS	P1	P2	L_c	Data	L_e
0x01	0x09	MSB Offset	LSB Offset	Data length	Attestation certificate	0x00

If attestation certificate will be transferred multiple parts to the TOE Parameter P1 and P2 defines the offset of the attestation certificate data chunk which will be transfer with the actual command APDU.

The data part contains the whole or only a part of attestation certificate. If the size of the attestation certificate exceeds the size of data that can be transferred in one APDU, the certificate can be splitted to multiple parts which can be transferred in separate APDUs.

Incoming attestation certificate data (parts) will be proceeded and stored in reserved internal storage. The storage size has to be set up as parameter at the installation process of the applet. This TSFI generate the TOEs private keys seed and MACKey after the storage is completely filled. After the keys are generated the TOE will switch to state READY_FOR_USE. Certificate bytes can be send in chunks of arbitrary size. As long as the storage is not completed filled the TOE will stay in state “uninitialized”. This function will not check if the stored private key matches to the received attestation certificate.

The Response APDU contains no data field. The SW will be 0x9000 for successful processing the Command APDU or 0x6A80 if the size of the of all previous data in the SET_ATTESTATION_CERT command APDUs plus the actual data field size exceeds the size that was defined at the installation process of the applet.

If the token is already in state READY_FOR_USE the token will reject the command with SW 0x6982.

This is a **SFR-non-interfering** interface because it controls and changes the internal state and triggers the key generation.

1.5 TSFI GET_RESPONSE

This TSFI is used to request remain response data from the TOE if response to one of the other commands didn’t fit in a single response APDU and no extended length was used.

CLA	INS	P1	P2	L_c	Data	L_e
0x00	0xC0	0x00	0x00	-	-	0x00 or data length to receive

The parameters P1 and P2 will not be checked and have no further function. They shall both be set to 0x00.

This APDU expects no data and will ignore incoming data.

If the Le field is set to 0x00, then all the available response data bytes of the previous command will be returned within the limit of 256 byte if no extended length is used. Any other value will return the requested size or less if the available size is smaller than the requested size.

This command is used for chaining (see [ISO 7816-4]) if extended length APDUs can't be used and more data have to transferred in response to a command then fit in one APDU. This function will indicate how much data is still available to get in the SW (see [ISO 7816-4] or FIDO U2F specification).

The response APDU will contain the requested response data in the data field. The SW will be 0x61xx where xx is the size of that data that's still available to receive with further GET RESPONSE calls. If xx is 0x00 then the maximum size of data (256 byte if no extended length was used) is available.

The response will be SW 0x6985 if this command was called without a previous command (U2F_REGISTER or U2F_AUTHENTICATE) that signals that there is data to fetch by returning SW 0x61xx.

Because this interface doesn’t perform any SFR relevant actions itself but supports commands which do perform SFR relevant actions this interface is SFR-supporting.

1.6 TSFI RESET

This TSFI is used to request a reset of the internal keys and the FIDO signing counter.

CLA	INS	P1	P2	L_c	Data	L_e
0x00	0x8E	0x5E	0x70	-	-	0x00

Parameter P1 has to be 0x5E and P2 has to be 0x70. Otherwise the command will not be executed.

This APDU expects no data and will ignore incoming data.

This function will check if user presence check is need, enforce it if needed and checks if the internal state is READY_FOR_USE. The keys seed and MACKey will be securely cleared. After successful clearing the keys the FIDO signing counter will be reset to zero and new keys for seed and MACKey will be generated by using a DRG.4 based random number generator.

The response APDU will contain no data field. The SW bytes will be

- SW 0x9000 if the reset was successful.
- SW 0x6A86 if the parameters P1 and P2 didn't match the expected bytes. The keys are kept untouched.
- SW 0x6200 if the reset failed. The keys are kept untouched.
- SW 0x6985 if the user presence check failed. The keys are kept untouched.

This is a **SFR-enforcing** interface because it is used to clear and regenerate the keys seed and MACKey after it performs the user presence check.

1.7 Generic status words

Beside the status words (SW) that are command specific and described within the TSFI above, the applet also may return the following generic SW:

- SW 0x6982 (Security conditions not fulfilled): Will be returned if proprietary CLA byte (0x01) was used, but the card is already in state DELIVERY_STATE or READY_TO_USE.
- SW 0x6D00 (Instruction not supported): Will be returned if proprietary CLA byte (0x01) was used and the card is in state UNINITIALIZED, but the INS byte wasn't set to 0x09 (Set attestation certificate).
  Will also be returned if CLA byte was 0x00, but the INS byte doesn't contain any of the known instructions specified in this document
- SW 0x6E00 (Class not supported): Will be returned if CLA byte was neither set to 0x00 nor 0x01.

2 SFR to TSFI Tracing

	TSFI U2F_REGISTER	TSFI U2F_AUTHENTICATE	TSFI U2F_GET_VERSION	TSFI SET_ATTESTATION_CERT	TSFI GET_RESPONSE	TSFI RESET
FDP_IFC.1/ Initialisation						x
FDP_IFF.1/ Initialisation						x
FDP_IFC.1/ Reset						x
FDP_IFF.1/ Reset						x
FDP_IFC.1/ Registration	x				x	
FDP_IFF.1/ Registration	x				x	
FDP_IFC.1/ Authentication		x			x	
FDP_IFF.1/ Authentication		x			x	
FMT_SMF.1						x
FMT_SMR.1	x	x				x
FMT_MTD.1	x	x			x	x
FMT_MSA.1/ Initialisation						x
FMT_MSA.3/ Initialisation						x
FMT_MSA.1/ Reset						x
FMT_MSA.3/ Reset						x
FMT_MSA.1/ Registration	x				x	
FMT_MSA.3/ Registration	x				x	
FMT_MSA.1/ Authentication		x			x	
FMT_MSA.3/ Authentication		x			x	

	TSFI U2F_REGISTER	TSFI U2F_AUTHENTICATE	TSFI U2F_GET_VERSION	TSFI SET_ATTESTATION_CERT	TSFI GET_RESPONSE	TSFI RESET
FCS_CKM.1						x
FCS_CKM.4						x
FCS_COP.1	x	x				
FCS_COP.1/ MAC	x	x				
FCS_RNG.1	x					x
FDP_SDI.1[1]						
FCS_CKM.5/ U2F-Private	x	x				
FCS_CKM.5/ U2F-Public	x					
FPR_ANO.1[2]						
FIA_UAU.2	x	x				x
FIA_UAU.6	x	x				x
FPT_EMS.1[3]						
FPT_PHP.3[4]						
FPT_TST.1[5]						

1 This SFR is implicitly satisfied by the underlying Java Card Platform
2 This SFR is implicitly satisfied by design of the FIDO U2F standard
3 This SFR is implicitly satisfied by the underlying hardware security platform
4 This SFR is implicitly satisfied by the underlying hardware security platform
5 This SFR is implicitly satisfied by the underlying Java Card Platform

Glossary

U2F Universal Second Factor

For further FIDO related terms see the „FIDO Technical Glossary“ of [FIDOSpec].

Reference Documentation

FIDO U2F Raw	Fido Alliance: FIDO U2F Raw Message Formats
ISO 7816-4	ISO/IEC: Identification cards - Integerated circuit cards, Part 4: Organization, security and commands for interchange
FIDOSpec	FIDO Alliance: Fido Alliance Universal 2nd Factor 1.1 Specifications