Federal Office
for Information Security

Developer Documentation ATE - Developer Tests

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

1.2 (13.11.2019)

Federal Office for Information Security
Post Box 20 03 63
D-53133 Bonn
Internet: https://www.bsi.bund.de

Table of Contents

Figures

Tables

1 Test Plan

1.1 Test environment

The test environment consists of a standard PC with Ubuntu 17.10 64 bit operating system and a contactless reader ReinerSCT cyberJack RFID basis. To perform the tests over the contact-based interface the card reader SCM Microsystems SRC531 is used. The readers are connected via USB interface and uses the PC/SC (CCID) driver from the OS. JUnit 4 is used as test framework in an eclipse Oxygen.2 Release 4.7.2 IDE. The test cases are implemented in standard Java code. The standard javax.smartcardio Java Package is used to control the smartcard reader and send APDUs to the TOE.

The test can be performed on any system, that can send and receive APDU commands. On desktop PCs mainly card readers which supports the PC/SC interface can communicate with the TOE. On mobile devices the NFC can be used to communicate with the TOE. The testcases defined herein are implemented as JUnit test cases. In principle the test cases could be performed with any tool that can send and receive APDU.

1.2 Test configurations

The Device Under Test (DUT) can be configured to run all test cases without checking the user presence for each test case. For that the de.fac2 applet can be installed with flag set to 0x01 at the installation procedure. The meaning of this flag is specified in [AGD] chapter 4 “Product Integrator Guidance”
A deactivated User Presence Check will not affect any other functionality of the DUT as this flag only bypasses the User Presence Check if it occurs in the protocol sequence. The User Presence Check has no further impact on any crypto or protocol handling. Test runs with configuration CFG_DEACTIVATED_USER_PRESENCE_CHECK will return the same results as test runs with configuration CFG_DEFAULT except that there is no need for remove the DUT from the reader of every test case.

If the test cases will be performed with configuration CFG_DEACTIVATED_USER_PRESENCE_CHECK the Test prerequisite step “Place DUT onto the reader” and the Test post processing step “Remove DUT from reader” shall be ignored. Although the DUT should be placed onto reader at least before the first test case and might be removed after the last test case.

Test configuration	Description
CFG_DEFAULT	User presence check is active. DUT has to be removed from reader after each test case.
CFG_DEACTIVATED_USER_PRESENCE_CHECK	User presence check is deactivated.

The following additional configuration have to be set

- No other applet is installed on the platform than the de.fac2 applet.
- Either the contact-based or contactless interface may be used. The used interface will be document in the test result.
- The platform is in GlobalPlatform card life cycle state SECURED [GP221]
- The applet has to be in READY_FOR_USE state. Therefore, a attestation certificate has to be installed on the TOE.

The attestation certificate on the DUT has no direct impact on any function of the DUT. For testing the DUT may contains a self generated attestation certificate. The final TOE in production will use a different attestation certificate.

1.3 Test cases

The herein defined test cases are based on the [U2F FIDO NFC Tests] which are implemented in C. For more flexibility and a unified workflow the tests were re-implemented in Java. Additional test cases were created to test the reset function which is not part of the FIDO U2F standard.

The following ISO7816-4 defined status words have a special meaning in U2F:

- SW_NO_ERROR (0x9000): The command completed successfully without error.
- SW_CONDITIONS_NOT_SATISFIED (0x6985): The request was rejected due to test-of-user-presence being required.
- SW_WRONG_DATA (0x6A80): The request was rejected due to an invalid key handle.
- SW_WRONG_LENGTH (0x6700): The length of the request was invalid.
- SW_CLA_NOT_SUPPORTED (0x6E00): The Class byte of the request is not supported.
- SW_INS_NOT_SUPPORTED (0x6D00): The Instruction of the request is not supported.

1.3.1 Test case U2F_APPLET_SELECT

Test case identification	U2F_APPLET_SELECT
Test case description	Check that the DUT responds with correct version string in response to select applet command
Test configuration	CFG_DEFAULT or CFG_DEACTIVATED_USER_PRESENCE_CHECK
Test environment	Default test environment as described in chapter 1.1
Test prerequisites	Place DUT onto the reader
Test procedure	Send Select APDU with U2F-AID: `'00 A4 04 00 08 A0 00 00 06 47 2F 00 01 00'`
Expected result	Receive U2F version string “U2F_V2” in data part of response APDU and status word 0x9000: `'55 32 46 5F 56 32 90 00'`
Actual result	See chapter 1.4 for test result
Test post processing	ATE_FUN.1-2 Remove DUT from reader
Remark	-

1.3.2 Test case U2F_UNKNOWN_INSTRUCTION

Test case identification	U2F_UNKNOWN_INSTRUCTION

Test case description	Check that the DUT responds with expected SW if an unknown instruction byte was sent in command APDU
Test configuration	CFG_DEFAULT or CFG_DEACTIVATED_USER_PRESENCE_CHECK
Test environment	Default test environment as described in chapter 1.1
Test prerequisites	Place DUT onto the reader

Test procedure	1. Send Select APDU with U2F-AID:
	`'00 A4 04 00 08 A0 00 00 06 47 2F 00 01 00'`
	2. Send APDU with unknown instruction byte 0x00:
	`'00 00 00 00 00'`

Expected result	1. Receive U2F version string “U2F_V2” in data part of response APDU and status word 0x9000: `'55 32 46 5F 56 32 90 00'` 2. Receive SW 0x6D00
Actual result	See chapter 1.4 for test result
Test post processing	Remove DUT from reader
Remark	-

1.3.3 Test case U2F_UNKNOWN_INSTRUCTION_EXTENDED_LENGTH

Test case identification	U2F_UNKNOWN_INSTRUCTION_EXTENDED_LENGTH
Test case description	Check that the DUT responds with expected SW if an unknown instruction byte was sent in command APDU using extend length format (Case 4 Extended)
Test configuration	CFG_DEFAULT or CFG_DEACTIVATED_USER_PRESENCE_CHECK
Test environment	Default test environment as described in chapter 1.1
Test prerequisites	Place DUT onto the reader

Test procedure	1. Send Select APDU with U2F-AID:
	`'00 A4 04 00 08 A0 00 00 06 47 2F 00 01 00'`
	2. Send APDU with unknown instruction byte 0x00:
	`'00 00 00 00 00 00 01 FF 00 00'`

Expected result	1. Receive U2F version string “U2F_V2” in data part of response APDU and status word 0x9000:

	'55 32 46 5F 56 32 90 00' 2. Receive SW 0x6D00
Actual result	See chapter 1.4 for test result
Test post processing	Remove DUT from reader
Remark	-

1.3.4 Test case U2F_BAD_CLASS

Test case identification	U2F_BAD_CLASS
Test case description	Check that the DUT responds with expected SW if an unknown class byte was sent in command APDU
Test configuration	CFG_DEFAULT or CFG_DEACTIVATED_USER_PRESENCE_CHECK
Test environment	Default test environment as described in chapter 1.1
Test prerequisites	Place DUT onto the reader
Test procedure	1. Send Select APDU with U2F-AID: '00 A4 04 00 08 A0 00 00 06 47 2F 00 01 00' 2. Send APDU with unknown class byte 0x01: '01 02 00 00 00'
Expected result	1. Receive U2F version string “U2F_V2” in data part of response APDU and status word 0x9000: '55 32 46 5F 56 32 90 00' 2. Receive SW other than 0x9000
Actual result	See chapter 1.4 for test result
Test post processing	Remove DUT from reader
Remark	-

1.3.5 Test case U2F_REGISTER_WRONG_LENGTH

Test case identification	U2F_REGISTER_WRONG_LENGTH
Test case description	Check that the DUT responds with expected SW if an U2F Register command APDU with wrong data length
Test configuration	CFG_DEFAULT or CFG_DEACTIVATED_USER_PRESENCE_CHECK
Test environment	Default test environment as described in chapter 1.1
Test prerequisites	Place DUT onto the reader

Test procedure	1. Send Select APDU with U2F-AID:
	`'00 A4 04 00 08 A0 00 00 06 47 2F 00 01 00'`
	2. Send APDU with wrong data length byte 0x00:
	`'00 01 00 00 00'`

Expected result	1. Receive U2F version string “U2F_V2” in data part of response APDU and status word 0x9000: `'55 32 46 5F 56 32 90 00'` 2. Receive SW 0x6700
Actual result	See chapter 1.4 for test result
Test post processing	Remove DUT from reader
Remark	-

1.3.6 Test case U2F_VALID_REGISTER

Test case identification	U2F_VALID_REGISTER
Test case description	Perform a U2F registration by sending U2F Register command APDU with valid registration data to DUT
Test configuration	CFG_DEFAULT or CFG_DEACTIVATED_USER_PRESENCE_CHECK
Test environment	Default test environment as described in chapter 1.1
Test prerequisites	Place DUT onto the reader

Test procedure	1. Send Select APDU with U2F-AID:
	`'00 A4 04 00 08 A0 00 00 06 47 2F 00 01 00'`
	2. Send U2F Register APDU:
	`‘00 01 00 00 40 <chp> <app> 00’`
	with ‘chp’ as the challenge parameter of 32 byte length and ‘app’ as the application parameter of 32 byte length.
	3. Verify the signature in the Registration Response Message

Expected result	1. Receive U2F version string “U2F_V2” in data part of response APDU and status word 0x9000:
	`‘55 32 46 5F 56 32 90 00’`
	2. Receive first data part and SW 0x6100. Send Get Response command APDU
	`‘00 C0 00 00 00’`
	and receive remaining data parts until SW was 0x9000. The received and reassembled data parts shall contain a valid Registration Response Message.
	3. Signature is correct

Actual result	See chapter 1.4 for test result
Test post processing	Remove DUT from reader
Remark	-

1.3.7 Test case U2F_VALID_REGISTER_SHORT_LE

Test case identification	U2F_VALID_REGISTER_SHORT_LE
Test case description	Perform a U2F registration by sending U2F Register command APDU with valid registration data to DUT. Using Le Byte with value 0x64 instead of default value 0x00.
Test configuration	CFG_DEFAULT or CFG_DEACTIVATED_USER_PRESENCE_CHECK
Test environment	Default test environment as described in chapter 1.1
Test prerequisites	Place DUT onto the reader

Test procedure	1. Send Select APDU with U2F-AID:
	`'00 A4 04 00 08 A0 00 00 06 47 2F 00 01 00'`
	2. Send U2F Register APDU:
	`‘00 01 00 00 40 <chp> <app> 64’`
	with ‘chp’ as the challenge parameter of 32 byte length and ‘app’ as the application parameter of 32 byte length.
	3. Verify the signature in the Registration Response Message

Expected result	1.Receive U2F version string “U2F_V2” in data part of response APDU and status word 0x9000:
	`‘55 32 46 5F 56 32 90 00’`
	2. Receive first data part and SW 0x6100. Send Get Response command APDU
	`‘00 C0 00 00 64’`
	and receive remaining data parts until SW was 0x9000. The received and reassembled data parts shall contain a valid Registration Response Message.
	3. Signature is correct

Actual result	See chapter 1.4 for test result
Test post processing	Remove DUT from reader
Remark	-

1.3.8 Test case U2F_VALID_REGISTER_EXTENDED_LENGTH

Test case identification	U2F_VALID_REGISTER_EXTENDED_LENGTH
Test case description	Perform a U2F registration by sending U2F Register command APDU with valid registration data to DUT. Using extended length APDU format (Case 4 Extended)
Test configuration	CFG_DEFAULT or

	CFG_DEACTIVATED_USER_PRESENCE_CHECK
Test environment	Default test environment as described in chapter 1.1
Test prerequisites	Place DUT onto the reader
Test procedure	1. Send Select APDU with U2F-AID: '00 A4 04 00 08 A0 00 00 06 47 2F 00 01 00' 2. Send U2F Register APDU: ‘00 01 00 00 00 00 40 <chp> <app> 09 8A’ with ‘chp’ as the challenge parameter of 32 byte length and ‘app’ as the application parameter of 32 byte length. 3. Verify the signature in the Registration Response Message
Expected result	1. Receive U2F version string “U2F_V2” in data part of response APDU and status word 0x9000: ‘55 32 46 5F 56 32 90 00’ 2. Receive SW 0x9000. The received data shall contain a valid Registration Response Message. 3. Signature is correct
Actual result	See chapter 1.4 for test result
Test post processing	Remove DUT from reader
Remark	-

1.3.9 Test case U2F_VALID_AUTHENTICATE

Test case identification	U2F_VALID_AUTHENTICATE
Test case description	Perform a U2F authentication by sending U2F Authenticate command APDU with valid Authentication Request Message to the DUT
Test configuration	CFG_DEFAULT or CFG_DEACTIVATED_USER_PRESENCE_CHECK
Test environment	Default test environment as described in chapter 1.1
Test prerequisites	1. Place DUT onto the reader 2. Perform test case U2F_VALID_REGISTER
Test procedure	1. Send Select APDU with U2F-AID: '00 A4 04 00 08 A0 00 00 06 47 2F 00 01 00' 2. Send U2F Authenticate APDU: ‘00 02 03 00 81 <chp> <app> 40 <kh> 00’ with ‘chp’ as the challenge parameter of 32 byte length and ‘app’ as the application parameter of 32 byte length both which were used at the previous registration process and ‘kh’ as the keyhandle received in response to the previous registration process 3. Verify the signature in the Authentication Response Message

Expected result	1. Receive U2F version string “U2F_V2” in data part of response APDU and status word 0x9000: ‘55 32 46 5F 56 32 90 00’ 2. Receive a valid Authentication Response Message with SW 0x9000 3. Signature is correct
Actual result	See chapter 1.4 for test result
Test post processing	Remove DUT from reader
Remark	-

1.3.10 Test case U2F_VALID_AUTHENTICATE_EXTENDED_LENGTH

Test case identification	U2F_VALID_AUTHENTICATE_EXTENDED_LENGTH
Test case description	Perform a U2F authentication by sending U2F Authenticate command APDU with valid Authentication Request Message to the DUT. Using extended length APDU format (Case 4 Extended)
Test configuration	CFG_DEFAULT or CFG_DEACTIVATED_USER_PRESENCE_CHECK
Test environment	Default test environment as described in chapter 1.1
Test prerequisites	1. Place DUT onto the reader 2. Perform test case U2F_VALID_REGISTER
Test procedure	1. Send Select APDU with U2F-AID: '00 A4 04 00 08 A0 00 00 06 47 2F 00 01 00' 2. Send U2F Authenticate APDU: ‘00 02 03 00 00 00 81 <chp> <app> 40 <kh> 00 4D’ with ‘chp’ as the challenge parameter of 32 byte length and ‘app’ as the application parameter of 32 byte length both which were used at the previous registration process and ‘kh’ as the keyhandle received in response to the previous registration process 3. Verify the signature in the Authentication Response Message
Expected result	1. Receive U2F version string “U2F_V2” in data part of response APDU and status word 0x9000: ‘55 32 46 5F 56 32 90 00’ 2. Receive a valid Authentication Response Message with SW 0x9000 3. Signature is correct
Actual result	See chapter 1.4 for test result
Test post processing	Remove DUT from reader
Remark	-

1.3.11 Test case U2F_WRONG_KEYHANDLE

Test case identification	U2F_WRONG_KEYHANDLE
Test case description	Perform a U2F authentication by sending U2F Authenticate command APDU with invalid Authentication Request Message to the DUT. Authentication Request shall contain a wrong keyHandle.
Test configuration	CFG_DEFAULT or CFG_DEACTIVATED_USER_PRESENCE_CHECK
Test environment	Default test environment as described in chapter 1.1
Test prerequisites	1. Place DUT onto the reader 2. Perform test case U2F_VALID_REGISTER
Test procedure	1. Send Select APDU with U2F-AID: '00 A4 04 00 08 A0 00 00 06 47 2F 00 01 00' 2. Send U2F Authenticate APDU: ‘00 02 03 00 81 <chp> <app> 40 <kh> 00’ with ‘chp’ as the challenge parameter of 32 byte length and ‘app’ as the application parameter of 32 byte length both which were used at the previous registration process and ‘kh’ as the keyhandle received in response to the previous registration process but modified first byte. E.g. keyHandle[0] XOR 0x55
Expected result	1. Receive U2F version string “U2F_V2” in data part of response APDU and status word 0x9000: ‘55 32 46 5F 56 32 90 00’ 2. Receive SW 0x6A80
Actual result	See chapter 1.4 for test result
Test post processing	Remove DUT from reader
Remark	-

1.3.12 Test case U2F_WRONG_APPID

Test case identification	U2F_WRONG_APPID
Test case description	Perform a U2F authentication by sending U2F Authenticate command APDU with invalid Authentication Request Message to the DUT. Authentication Request shall contain a wrong AppId. Using extended length APDU format (Case 4 Extended)
Test configuration	CFG_DEFAULT or CFG_DEACTIVATED_USER_PRESENCE_CHECK

Test environment	Default test environment as described in chapter 1.1
Test prerequisites	1. Place DUT onto the reader 2. Perform test case U2F_VALID_REGISTER

Test procedure	1. Send Select APDU with U2F-AID:
	`'00 A4 04 00 08 A0 00 00 06 47 2F 00 01 00'`
	2. Send U2F Authenticate APDU:
	`‘00 02 03 00 00 00 81 <chp> <app> 40 <kh> 00 4D’`
	with ‘chp’ as the challenge parameter of 32 byte length and ‘app’ as the application parameter of 32 byte length both which were used at the previous registration process. The value of ‘app’ shall be modified e.g by using app[0] XOR 0xAA. Use ‘kh’ as the keyhandle received in response to the previous registration process.

Expected result	1. Receive U2F version string “U2F_V2” in data part of response APDU and status word 0x9000: `‘55 32 46 5F 56 32 90 00’` 2. Receive SW 0x6A80
Actual result	See chapter 1.4 for test result
Test post processing	Remove DUT from reader
Remark	-

1.3.13 Test case ADD_RESET

Test case identification	ADD_RESET
Test case description	Check that Authentication with previous valid authentication data fails after DUT has received Reset command APDU.
Test configuration	CFG_DEFAULT or CFG_DEACTIVATED_USER_PRESENCE_CHECK
Test environment	Default test environment as described in chapter 1.1
Test prerequisites	Place DUT onto the reader

Test procedure	1. Send Select APDU with U2F-AID:
	`'00 A4 04 00 08 A0 00 00 06 47 2F 00 01 00'`
	2. Send U2F Register APDU:
	`‘00 01 00 00 40 <chp> <app> 00’`
	with ‘chp’ as the challenge parameter of 32 byte length and ‘app’ as the application parameter of 32 byte length.
	2a. If test configuration CFG_DEFAULT is used, remove and replace DUT to reader and send Select APDU with U2F-AID:

	'00 A4 04 00 08 A0 00 00 06 47 2F 00 01 00' 3. Send U2F Authenticate APDU: ‘00 02 03 00 81 <chp> <app> 40 <kh> 00’ with ‘chp’ as the challenge parameter of 32 byte length and ‘app’ as the application parameter of 32 byte length both which were used at the previous registration process and ‘kh’ as the keyhandle received in response to the previous registration process 3a. If test configuration CFG_DEFAULT is used, remove and replace DUT to reader and send Select APDU with U2F-AID: '00 A4 04 00 08 A0 00 00 06 47 2F 00 01 00' 4. Send Reset command APDU: ‘00 8E 5E 70 00’ 4a. If test configuration CFG_DEFAULT is used, remove and replace DUT to reader and send Select APDU with U2F-AID: '00 A4 04 00 08 A0 00 00 06 47 2F 00 01 00' 5. Send U2F Authenticate APDU: ‘00 02 03 00 81 <chp> <app> 40 <kh> 00’ with same data as used in step 3.
Expected result	1. Receive U2F version string “U2F_V2” in data part of response APDU and status word 0x9000: ‘55 32 46 5F 56 32 90 00’ 2. Receive SW 0x9000. The received data shall contain a valid Registration Response Message. 2.a Receive U2F version string “U2F_V2” in data part of response APDU and status word 0x9000. 3. Receive a valid Authentication Response Message with SW 0x9000 3.a Receive U2F version string “U2F_V2” in data part of response APDU and status word 0x9000. 4. Receive SW 0x9000 4.a Receive U2F version string “U2F_V2” in data part of response APDU and status word 0x9000. 5. Receive SW 0x6A80
Actual result	See chapter 1.4 for test result
Test post processing	Remove DUT from reader
Remark	-

1.3.14 Test case ADD_USER_PRESENCE_CHECK_REGISTRATION

Test case identification	ADD_USER_PRESENCE_CHECK_REGISTRATION
Test case description	Check that DUT will enforce the user presence check for U2F Registration.
Test configuration	CFG_DEFAULT

Test environment	Default test environment as described in chapter 1.1
Test prerequisites	Place DUT onto the reader
Test procedure	1. Send Select APDU with U2F-AID: '00 A4 04 00 08 A0 00 00 06 47 2F 00 01 00' 2. Send U2F Register APDU: ‘00 01 00 00 40 <chp> <app> 00’ with ‘chp’ as the challenge parameter of 32 byte length and ‘app’ as the application parameter of 32 byte length. 3. Send again a U2F Register APDU: ‘00 01 00 00 40 <chp> <app> 00’ with ‘chp’ as the challenge parameter of 32 byte length and ‘app’ as the application parameter of 32 byte length. Use different values for ‘chp’ and ‘app’ as in step 2
Expected result	1. Receive U2F version string “U2F_V2” in data part of response APDU and status word 0x9000: ‘55 32 46 5F 56 32 90 00’ 2. Receive SW 0x9000. The received data shall contain a valid Registration Response Message. 3. Receive SW 0x6985
Actual result	See chapter 1.4 for test result
Test post processing	Remove DUT from reader
Remark	-

1.3.15 Test case ADD_USER_PRESENCE_CHECK_AUTHENTICATION

Test case identification	ADD_USER_PRESENCE_CHECK_AUTHENTICATION
Test case description	Check that DUT will enforce the user presence check for U2F Authentication.
Test configuration	CFG_DEFAULT
Test environment	Default test environment as described in chapter 1.1
Test prerequisites	Place DUT onto the reader
Test procedure	1. Send Select APDU with U2F-AID: '00 A4 04 00 08 A0 00 00 06 47 2F 00 01 00' 2. Send U2F Register APDU: ‘00 01 00 00 40 <chp> <app> 00’ with ‘chp’ as the challenge parameter of 32 byte length and ‘app’ as the application parameter of 32 byte length. 3. Send U2F Authenticate APDU: ‘00 02 03 00 81 <chp> <app> 40 <kh> 00’ with ‘chp’ as the challenge parameter of 32 byte length and ‘app’ as the

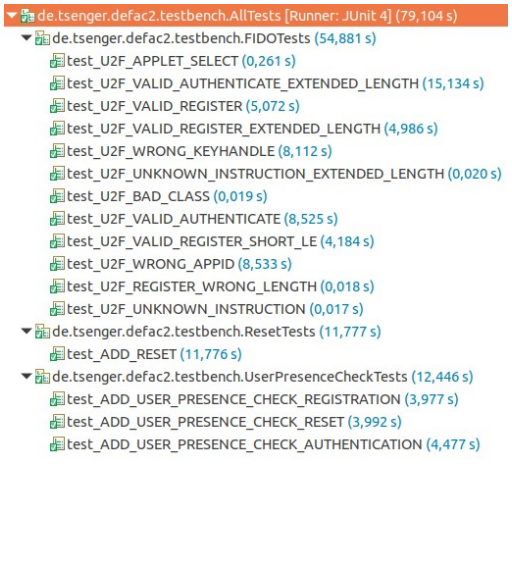
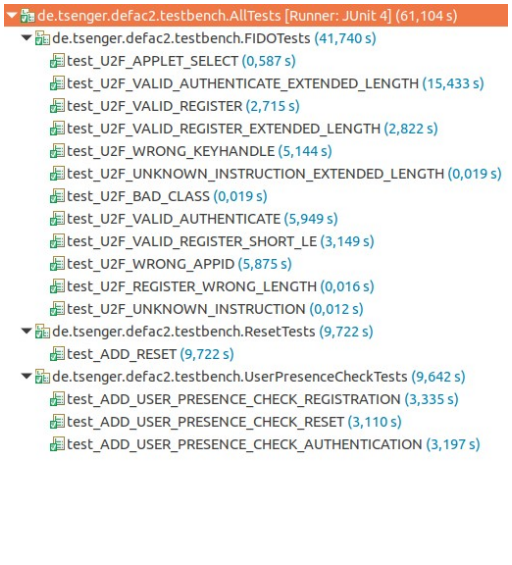
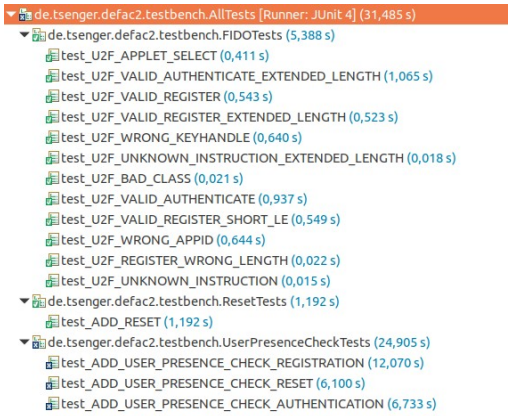
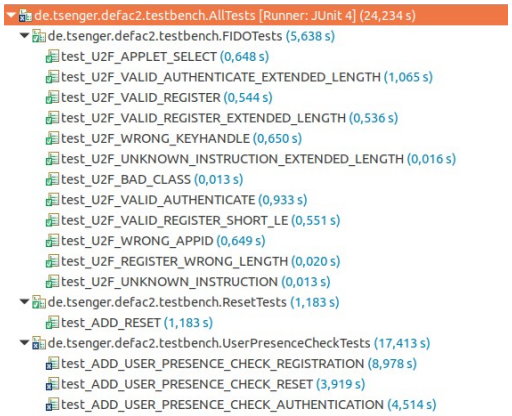
	application parameter of 32 byte length both which were used at the previous registration process and ‘kh’ as the keyhandle received in response to the previous registration process
Expected result	1. Receive U2F version string “U2F_V2” in data part of response APDU and status word 0x9000: ‘55 32 46 5F 56 32 90 00’ 2. Receive SW 0x9000. The received data shall contain a valid Registration Response Message. 3. Receive SW 0x6985
Actual result	See chapter 1.4 for test result
Test post processing	Remove DUT from reader
Remark	-

1.3.16 Test case ADD_USER_PRESENCE_CHECK_RESET

Test case identification	ADD_USER_PRESENCE_CHECK_RESET
Test case description	Check that DUT will enforce the user presence check for Reset process.
Test configuration	CFG_DEFAULT
Test environment	Default test environment as described in chapter 1.1
Test prerequisites	Place DUT onto the reader
Test procedure	1. Send Select APDU with U2F-AID: '00 A4 04 00 08 A0 00 00 06 47 2F 00 01 00' 2. Send U2F Register APDU: ‘00 01 00 00 40 <chp> <app> 00’ with ‘chp’ as the challenge parameter of 32 byte length and ‘app’ as the application parameter of 32 byte length. 3. Send Reset command APDU: ‘00 8E 5E 70 00’
Expected result	1. Receive U2F version string “U2F_V2” in data part of response APDU and status word 0x9000: ‘55 32 46 5F 56 32 90 00’ 2. Receive SW 0x9000. The received data shall contain a valid Registration Response Message. 3. Receive SW 0x6985
Actual result	See chapter 1.4 for test result
Test post processing	Remove DUT from reader
Remark	-

1.4 Overall test result

All test cases from chapter 1.3 were performed at 2019-03-07 once in test configuration CFG_DEACTIVATED_USER_PRESENCE_CHECK and once in test configuration CFG_DEFAULT. Both test configuration also were performed once on the contact-based reader and once on the contactless reader. So all testcases were performed four times in different combination of reader interfaces and test configuration.

		Reader Interface	
		contactless	contact-based
Test Configuration	CFG_DEFAULT	 All test cases passed	 All test cases passed
	CFG_DEACTIVATED_USER_PRESENCE_CHECK	 All test cases passed except the 3 user presence tests	 All test cases passed except the 3 user presence tests

The test cases ADD_USER_PRESENCE_CHECK_REGISTRATION, ADD_USER_PRESENCE_CHECK_AUTHENTICATION and ADD_USER_PRESENCE_CHECK_RESET can only be performed in test configuration CFG_DEFAULT because they explicit test the function that is disabled in test configuration CFG_DEACTIVATED_USER_PRESENCE_CHECK.

The overall test result is “pass”. The test results have been recorded in the following test protocols:

- 2019-03-07_CFG_DEACTIVATED_USER_PRESENCE_CHECK_contact-based.log
- 2019-03-07_CFG_DEACTIVATED_USER_PRESENCE_CHECK_contactless.log
- 2019-03-07_CFG_DEFAULT_contact-based.log
- 2019-03-07_CFG_DEFAULT_Contactless.log

2 Analysis of Test Coverage - TSFI

The Functional Specification [de.fac2 FSP] defines six interfaces as TSFI. The following table demonstrates the correspondence between the tests in the test documentation and the TSFI in the FSP.

TSFI	Mapped SFRs	Tests	Analysis
U2F_REGISTER	FDP_IFC.1/ Registration FDP_IFF.1/ Registration FMT_SMR.1 FMT_MTD.1 FMT_MSA.1/ Registration FMT_MSA.3/ Registration FCS_COP.1 FCS_COP.1/ MAC FCS_RNG.1/ nonce FCS_CKM.5/ U2F-Private FCS_CKM.5/ U2F-Public FIA_UAU.2 FIA_UAU.6	U2F_REGISTER_WRONG_LENGTH U2F_VALID_REGISTER U2F_VALID_REGISTER_SHORT_LE U2F_VALID_REGISTER_EXTENDED_LENGTH ADD_USER_PRESENCE_CHECK_REGISTRATION U2F_UNKNOWN_INSTRUCTION U2F_UNKNOWN_INSTRUCTION_EXTENDED_LENGTH U2F_BAD_CLASS	The test results demonstrate that the TSFI U2F REGISTER acts like expects. It could be shown that the registration fails if the input parameter doesn’t match the expected values and performs a valid registration with expected output for different valid input parameters. The enforcement of the user presence check worked liked expected. Unknown class byte or unknown instruction byte are resulted in an expected error response.

U2F_AUTHENTICATE	FDP_IFC.1/ Authentication FDP_IFF.1/ Authentication FMT_SMR.1 FMT_MTD.1 FMT_MSA.1/ Authentication FMT_MSA.3/ Authentication FCS_COP.1 FCS_COP.1/ MAC FCS_CKM.5/ U2F-Private FIA_UAU.2 FIA_UAU.6	U2F_VALID_AUTHENTICATE U2F_VALID_AUTHENTICATE_EXTENDED_LENGTH U2F_WRONG_KEYHANDLE U2F_WRONG_APPID ADD_USER_PRESENCE_CHECK_AUTHENTICATION U2F_UNKNOWN_INSTRUCTION U2F_UNKNOWN_INSTRUCTION_EXTENDED_LENGTH U2F_BAD_CLASS	The test results demonstrate that the TSFI U2F_AUTHENTICATE acts like expects. It could be shown that the authentication fails if either the keyhandle or the appId contains unregistered or unknown values. The authentication returns expected output values with a valid signature for different valid input parameters. The enforcement of the user presence check worked liked expected. Unknown class byte or unknown instruction byte are resulted in an expected error response.
U2F_GET_VERSION	- SFR non-interfering -	U2F_APPLET_SELECT	There is no explicit test case for this TSFI as this TSFI is categorized as SFR non-interfering. The test case U2F_APPLET_SELECT also return the U2F version and can be used to ensure, that the TOE returns the expected value.
SET_ATTESTATION_CERT	- SFR non-interfering -	-	There is no explicit test case for this TSFI as this TSFI is categorized as SFR non-interfering and not accessible in the card status READY_FOR_USE.

GET_RESPONSE	FDP_IFC.1/ Registration FDP_IFF.1/ Registration FDP_IFC.1/ Authentication FDP_IFF.1/ Authentication FMT_MTD.1 FMT_MSA.1/ Registration FMT_MSA.3/ Registration FMT_MSA.1/ Authentication FMT_MSA.3/ Authentication	U2F_VALID_REGISTER U2F_VALID_REGISTER_SHORT_LE U2F_VALID_AUTHENTICATE	The TSFI GET_RESPONSE is used in the referenced test cases to get response from the register and authentication commands. The test results demonstrate that the TSFI GET_RESPONSE acts like expects and the responses to this command are valid.
RESET	FDP_IFC.1/ Initialisation FDP_IFF.1/ Initialisation FDP_IFC.1/ Reset FDP_IFF.1/ Reset FMT_SMF.1 FMT_SMR.1 FMT_MTD.1 FMT_MSA.1/ Initialisation FMT_MSA.3/ Initialisation FMT_MSA.1/ Reset FMT_MSA.3/ Reset FCS_CKM.1 FCS_CKM.4 FCS_RNG.1/ nonce FIA_UAU.2 FIA_UAU.6	ADD_RESET ADD_USER_PRESENCE_CHECK_RESET	The test results demonstrate that the TSFI RESET acts like expects. It could be shown that a previously valid authentication attempt fails after a reset command. The enforcement of the user presence check worked liked expected.

3 Analysis of Test Depth – Subsystems

The Design Documentation [TDS] defines three subsystems and three modules. The following table demonstrates the correspondence between the tests in the test documentation and the TSF subsystems in the TOE design.

Subsystems	Modules	Tests	Analysis
Control & Communication subsystem	U2FApplet	• U2F_APPLET_SELECT (behaviour test) • U2F_UNKNOWN_INSTRUCTION (behaviour test) • U2F_UNKNOWN_INSTRUCTION_EXTENDED_LENGTH (behaviour test) • U2F_BAD_CLASS (behaviour test) • U2F_REGISTER_WRONG_LENGTH (behaviour test) • U2F_VALID_REGISTER(behaviour & interactions test) • U2F_VALID_REGISTER_SHORT_LE (behaviour & interactions test) • U2F_VALID_REGISTER_EXTENDED_LENGTH (behaviour & interactions test) • U2F_VALID_AUTHENTICATE (behaviour & interactions test) • U2F_VALID_AUTHENTICATE_EXTENDED_LENGTH (behaviour & interactions test) • U2F_WRONG_KEYHANDLE (behaviour & interactions test) • U2F_WRONG_APPID (behaviour & interactions test) • ADD_RESET (behaviour & interactions test) • ADD_USER_PRESENCE_CHECK_REGISTRATION (behaviour & interactions test) • ADD_USER_PRESENCE_CHECK_AUTHENTICATION (behaviour & interactions test) • ADD_USER_PRESENCE_CHECK_RESET (behaviour & interactions test)	The Control & Communication subsystem is involved in every single test case. All incoming commands are processed in the interface process and are then be forwarded to the other interfaces in this subsystem. Any error in this subsystem would lead to an expected behavior and a failed test result. Because all test cases are passed it is shown that this subsystem acts like expected.

Subsystems	Modules	Tests	Analysis
Crypto subsystem	FIDO-U2F-Implementation Curve Parameters	• U2F_VALID_REGISTER (behaviour & interactions test) • U2F_VALID_REGISTER_SHORT_LE (behaviour & interactions test) • U2F_VALID_REGISTER_EXTENDED_LENGTH (behaviour & interactions test) • U2F_VALID_AUTHENTICATE (behaviour & interactions test) • U2F_VALID_AUTHENTICATE_EXTENDED_LENGTH (behaviour & interactions test) • U2F_WRONG_KEYHANDLE (behaviour & interactions test) • U2F_WRONG_APPID (behaviour & interactions test) • ADD_RESET (behaviour & interactions test)	The responses to the referenced test cases contain can show that the crypto subsystem acts like expected. In case with an expected valid response it could be demonstrated that the applet creates valid public keys, key handles and signatures. The signature could be verified. In the test case where the keyhandle or appId was modified it could be demonstrated that the MAC calculation worked as expected as it rejected the incoming data. With the reset test, it could be demonstrated that the applet generates new internal keys so that previous registrations get invalid.
JavaCard platform	-	• All testcases (behaviour & interactions test)	The JavaCard platform subsystem is involved in every single test case as it provides every functionality the applet needs to work like expected.

Glossary

U2F Universal Second Factor

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

Reference Documentation

AGD	BSI: Developer Documentation AGD - User Guidance for de.fac2 Fido Authenticator Applet
GP221	GlobalPlatform: GlobalPlatform Card Specification Version 2.2.1
U2F FIDO NFC Tests	Google: Basic U2F NFC Test Harness at github: https://github.com/google/u2f-ref-code/tree/master/u2f-tests/NFC
de.fac2 FSP	Bundesamt für Sicherheit in der Informationstechnik: de.fac2 Developer Documentation FSP - Functional Specification
TDS	Bundesamt für Sicherheit in der Informationstechnik : de.fac2 - Developer Documentation TDS - TOE Design Description
FIDOSpec	FIDO Alliance: Fido Alliance Universal 2nd Factor 1.1 Specifications