MACHINE READABLE TRAVEL DOCUMENTS

TECHNICAL REPORT

RF PROTOCOL AND APPLICATION TEST STANDARD FOR EMRTD - PART 3

TESTS FOR APPLICATION PROTOCOL AND LOGICAL DATA STRUCTURE

Version: **V2.07**
Date – October 10, 2014

Published by authority of the Secretary General

File	: TR - RF and Protocol Testing Part 3 V2.07.doc
Author	: ISO/JTC1/SC17/WG3/TF4 for ICAO-NTWG

INTERNATIONAL CIVIL AVIATION ORGANIZATION

File	: TR - RF and Protocol Testing Part 3 V2.07.doc
Author	: ISO/JTC1/SC17/WG3/TF4 for ICAO-NTWG

Release Control

Release	Date	Description
0.1	23-11-2005	First draft based on the German WG3 TF4 contribution "eMRTD Conformity Testing" version 1.02 presented at TF4 meeting in Paris Nov 21-23, 2005
0.2	21-12-2005	Updated version with new ICAO TR layout.
0.9	17-03-2006	Changes according to resolved comments from the WG3 TF4 meeting in Ottawa, Jan 30 – Feb 02, 2006. The following major changes have been introduced: • Less restrictive verification of status words • Introduction of profiles to be tested
0.95	2006-08-31	Intermediate draft, updated version with resolved comments from the Graz meeting Jun 12-13, 2006 and editorial changes. Some comments from Graz still unresolved. Test suites D and E have been modified as follows: There will be one test suite to check the protected command and one to check the unprotected commend. Test suites D and E will no longer check the correctness of the SM implementation since this is handled in test suite C New test suites F and G for testing unprotected SelectFile and ReadBinary. Test suites D and E will check the protected SelectFile and ReadBinary commands respectively. Redefined test cases (proposal) for test unit LDS_B – tests for DG1 – MRZ Redefined test cases E_1 – E_3 and E_5 – E_22 because of unspecified EOF reading.
0.96	2006-11-29	Final internal draft version including all resolved comments from the Bled meeting, Oct 25-26, 2006. Editorial changes in test suites C_9 and C_11 to clarify the encoding of offsets with tag 54. Editorial changes in test suites B and C. Former tests C_20 to C_36 have been moved to test suite B because these tests cover security condition tests.
1.0	2006-12-18	Editorial changes in 7816_C_8, C_9, and C_11 (sequence of steps to be performed) and in LDS_D_4 (reference changed to DOC9303, Annex A3.2)
1.01	2007-02-20	Test case 7816_C_16: verification of Postconditions removed
2.00 RC1	2013-02-14	Integration of contribution *AFNOR & BSI Contribution – SAC & AA conformity tests v1.6, January 13, 2013.*
2.00 RC2	2013-02-15	Update version of TR-03111 in [R8]
2.01	2013-02-28	ICAO submission
2.02 RC1	2013-08-02	Comments resolution
2.02 RC2	2013-09-05	Editorial modification in subclause 2.2
2.02 RC3	2013-10-01	Modification after comments resolution during Singapore TF4R meeting
2.03	2013-11-19	ICAO submission
2.04 RC1	2013-12-17	Comments resolution
2.04 RC2	2013-12-20	Review of comments resolution

2.05 RC1	2014-02-13	Comment on ISO7816_P_75 resolution
2.06	2014-03-10	ICAO submission
2.07 RC1	2014-10-10	Comments resolution after eMRTD Madrid event and Salamanca WG3

Table of contents

1 Introduction

1.1 Scope and purpose

An essential element of the new ICAO compliant eMRTD is the addition of a Secure Contactless Integrated Circuit (SCIC) that holds securely biometric data of the eMRTD bearer within the ICAO defined Logical Data Structure (LDS).

Successful integration of the SCIC into the eMRTD depends upon active international cooperation between many companies and organizations.

The eMRTD has been specified and designed to operate correctly across a wide variety of reading infrastructures worldwide. The risk profile for the eMRTD indicates a high impact if that design includes a widespread error or fault. Therefore it is essential, that all companies and organizations involved make all reasonable efforts to minimize the probability that this error or fault remains undetected before that design is approved and eMRTDs are issued.

This test specification covers the application interface, i.e. the ISO7816 conformance of the eMRTD Chip and the conformance of the LDS.
The ISO7816 conformance tests are restricted to the commands defined in the LDS and PKI specifications ([R1] and [R2]) and Supplemental Access Control for Machine Readable Travel Documents specifications [R7]. Other commands especially file creation and personalization commands are beyond the scope of this document.

The logical data structure test layer analyses the encoding of the LDS objects stored on an eMRTD. This layer contains several test units, one for each LDS object (DG 1 - 16, EF.COM and EF.SOD). Another test unit verifies the integrity and consistency of the different data structures. The tests specified for this layer can be performed using a regular eMRTD or with given input data from a different source (e.g. file). The test configuration document specifies the source of the data.

Note that this test specification addresses functional aspects only. Security features is out of scope.

1.2 Assumptions

It is assumed that the electrical interface and the underlying transport protocol are functionally tested. Thus, failures introduced by the RF protocol are out of scope of the test cases defined here.

1.3 Terminology

The key words "MUST", "SHALL", "REQUIRED", "SHOULD", "RECOMMENDED", and "MAY" in this document are to be interpreted as described in [R3].

MUST	This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.
MUST NOT	This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification.
SHOULD	This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.

SHOULD NOT This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

MAY This word, or the adjective "OPTIONAL", mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

1.4 Glossary

Command data field	The command data field defines the data of a command APDU that follows the command header and the Lc field – except the Le field. Its length is defined by Lc.
Command header	The command header comprises the first four bytes of the command APDU sent to the eMRTD in compliance with ISO 7816-4 [R5]. The header consists of the bytes CLA, INS, P1, and P2.
Lc	Length field of the command APDU encoding the number of bytes in the command data field. In this specification, Lc is encoded in one byte (short length).
Le	Length field of the command APDU encoding the maximum number of bytes expected in the response data field. In this specification, Le is encoded in one byte (short length).
Response data	Response data is the string of bytes that is encoded in the response data field.
Response data field	The response data field defines the data – except the response trailer – that the eMRTD returns in the response APDU in compliance with ISO 7816-4 [R5].
Response trailer	The response trailer defines the last two bytes that the eMRTD returns in the response APDU. The response trailer consists of two status bytes in compliance with ISO 7816-4 [R5].
Status bytes	The status bytes SW1-SW2 indicate the processing state of the LDS application in compliance with ISO 7816-4 [R5].
‘80’, ‘AB CD’	Bytes or byte strings encoded in Hex-ASCII will be denoted in apostrophes.

1.5 Abbreviations

Abbreviation	
AA	Active authentication
AID	Application identifier
APDU	Application protocol data unit
AT	Authentication template
BAC	Basic access control
CA	Chip Authentication
CLA	Class byte
DF	Dedicated file
DG	Data group
DH	Diffie-Hellman
DO	Data object
EAC	Extended access control
ECDH	Elliptic Curve Diffie-Hellman
EF	Elementary file
FID	File identifier
INS	Instruction byte
KAEG	Key Agreement ElGamal-type
KAT	Key Agreement Template
LDS	Logical data structure
MRZ	Machine-readable zone
OID	Object identifier
P1, P2	Parameter bytes
PACE	Password Authenticated Connection Establishment
PACE v2	Password protocol referring the "Generic Mapping" and the "Integrated Mapping"
PCD	Proximity coupling device
PICC	Proximity integrated circuit card
PKD	Public-key directory
PKI	Public-key infrastructure
RF	Radio frequency
SAC	Supplemental Access Control
SCIC	Secure contactless integrated circuit
SFI	Short file identifier

Abbreviation	
SM	Secure Messaging
SOD	Security data object
SW1, SW2	Status bytes
TBD	To be defined
TLV	Tag, length, value

1.6 Reference documentation

The following documentation served as reference for this technical report:

[R1] *ICAO Doc 9303 “Machine Readable Travel Documents“*
[R2] *ICAO Technical Report “LDS and PKI Maintenance“, version 1.0, May 2011.*
[R3] *RFC 2119, S. Bradner, "Key Words for Use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997*
[R4] *ICAO Supplement to Doc 9303, Release 12. April 04, 2013.*
[R5] *ISO/IEC 7816-4:2005. Identification cards -- Integrated circuit cards -- Part 4: Organization, security and commands for interchange.*
[R6] *ISO/IEC 19794-5:2005. Information technology -- Biometric data interchange formats -- Part 5: Face image data.*
[R7] *Technical Report - Supplemental Access Control for Machine Readable Travel Documents version 1.01, November 2010*
[R8] *BSI TR-03111, Elliptic Curve Cryptography (ECC) Version 2.0, June 28, 2012*

2 General test requirements

The tests in this layer require a fully personalized eMRTD. This means that all mandatory data groups MUST be present.

This layer tests all mandatory ISO 7816 commands of the SCIC. There are additional test units for tests for optional features like BAC, PACE v2 and AA. Throughout this document, the term PACE used in profile definition refers to PACE v2 as defined in the specification [R7].

All tests are mandatory unless marked as optional or conditional.

2.1 Test setup

For setting up these tests, any contactless reader supporting type A and type B protocols can be used. One personalized eMRTD sample is needed for executing the tests.

2.2 Implementation conformance statement

In order to set up the tests properly, an applicant SHALL provide the information specified in Table 1 below.

The ICAO specification defines several optional elements that an eMRTD can support. This includes security mechanisms like BAC, PACE v2 and AA as well as additional data groups (DG 3 to DG 16). Since these elements are optional, it is not possible to define the corresponding tests as mandatory for each eMRTD. Therefore, this document specifies a set of profiles. Each profile covers a specific optional element. A tested eMRTD MUST be assigned to the supported profiles in the implementation conformance statement, and a test MUST only be performed if the eMRTD belongs to this profile. The ICAO profile contains the mandatory feature set for ICAO compliant eMRTDs. Therefore, this profile and its tests are mandatory for all eMRTDs.

Note there are no profile ID’s explicitly defined for DG 14 and DG 15 because the PACE, AA and EAC profiles cover these data groups implicitly as described below:
- DG15 is present in case of AA
- DG14 is present in case of PACE, EAC or AA-ECDSA.

Table 1: Test precondition table "Information on the product"

Information for test setup	**Profile**	**Applicant declaration**
Access control applied: • Plaintext • Basic Access Control • Extended Access Control • Supplemental Access Control – PACE v2	Plain BAC EAC PACE PACE-CAN PACE-DH PACE-ECDH PACE-IM PACE-GM	
LDS Version	ICAO	
Read Binary with odd instruction byte supported	OddIns	

Information for test setup	**Profile**	**Applicant declaration**
eMRTD contains elementary file with LDS Data Group 3	DG3	
eMRTD contains elementary file with LDS Data Group 4	DG4	
eMRTD contains elementary file with LDS Data Group 5	DG5	
eMRTD contains elementary file with LDS Data Group 6	DG6	
eMRTD contains elementary file with LDS Data Group 7	DG7	
eMRTD contains elementary file with LDS Data Group 8	DG8	
eMRTD contains elementary file with LDS Data Group 9	DG9	
eMRTD contains elementary file with LDS Data Group 10	DG10	
eMRTD contains elementary file with LDS Data Group 11	DG11	
eMRTD contains elementary file with LDS Data Group 12	DG12	
eMRTD contains elementary file with LDS Data Group 13	DG13	
eMRTD contains elementary file with LDS Data Group 16	DG16	
Authentication supported: • Passive Authentication • Active Authentication	ICAO AA AA-RSA AA-ECDSA	
MRZ provided with the samples	ICAO	
Country signing certificate	ICAO	
Document signer certificate if not contained in SOD	ICAO	
Expected value for document type (2 characters)	ICAO	
Configuration list described in the EF.CardAccess	PACE	*(Algorithm + Domain parameters)*
Invalid key reference for PACE v2 (used in test case ISO7816_P_09)	PACE	
Invalid password identifier for PACE v2 (used in test case ISO7816_P_08)	PACE	
valid PACE OID not supported by the eMRTD (used in test case ISO7816_P_68. If such an OID can't be provided, ISO7816_P_68 is not applicable)	PACE	
Command to send to the eMRTD to verify the chip's ability to still require Secured APDU after performing valid or incomplete PACE v2 protocol. If not provided, use '00 B0 81 00 00'.	PACE	

The test cases reference all profiles which define a precondition for the test execution. Therefore, “BAC, DG3” and “BAC, DG4” refer to eMRTD which protect DG3 and DG4 with BAC respectively. “BAC, EAC, DG3” and “BAC, EAC, DG4” refer to eMRTD which protect DG3 and DG4 with BAC and EAC respectively. , “PACE, DG3” and “PACE, DG4” refer to eMRTD which protect DG3 and DG4 with PACE respectively. “PACE, EAC, DG3” and “PACE, EAC, DG4” refer to eMRTD which protect DG3 and DG4 with PACE and EAC respectively.

2.3 Verification of ISO 7816-4 status bytes

For most of test cases defined in this document, the status bytes returned by the eMRTD are not exactly defined in the ICAO specification. In these cases the result analysis uses the scheme defined in the ISO 7816-4 [R5] in order to specify the expected result. It is only checked that the response belongs to the specified category. In cases where the expected result is unambiguously defined in the ICAO specification, the exact value is specified in the test case.

Proprietary status bytes outside the range of defined ISO status bytes will be treated as failures in the test cases.

Table 2: ISO 7816-4 status bytes

Status bytes category	Category name	Valid value range	Process behavior
Normal processing	Normal processing status bytes	‘90 00’ ‘61 XX’	Process completed
Warning processing	ISO warning	‘62 XX’ ‘63 XX’	Process completed
Execution error	ISO execution error	‘64 XX’with XX='00' or XX > ‘80’ ‘65 XX’ ‘66 XX’	Process aborted
Checking error	ISO checking error	‘67 XX’ ‘68 XX’ ‘69 XX’ ‘6A XX’ ‘6B XX’ ‘6C XX’ ‘6D XX’ ‘6E XX’ ‘6F XX’	Process aborted

Note 1: There is a significant difference between normal and warning processing on the one side and execution and checking error on the other side. The first group is returned if the process has been fully completed, and the eMRTD MAY return some additional data. The “process aborted” categories are issued if the command cannot be performed. Therefore, response data MUST NOT be returned. In all test cases where an execution or checking error is expected, it MUST be verified that the eMRTD does not return any response data except SM protocol elements (DO ‘99’ / ‘8E’).
Note 2: ‘64 01’to ‘64 80’ are excluded of ISO execution error list for eMRTD.

3 Security and Command Tests

3.1 Unit Test ISO_7816_A – SelectApplication Command

This test unit covers all tests about the SelectApplication command. The LDS specification requires the selection of the LDS application by its AID. Since the AID is unique, selecting the application SHOULD be possible regardless of the previously selected DF or EF. Selecting the LDS Application SHOULD also reset the cards security state but this scenario is tested in the access control unit test.

3.1.1 Test Case 7816_A_1

Purpose	Selecting the LDS Application using the AID (positive test)
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	ICAO, Plain
Preconditions	LDS application MUST NOT be selected.
Test scenario	1. Send the following SelectApplication APDU to the eMRTD. => ‘00 A4 04 0C 07 A0 00 00 02 47 10 01’
Expected results	1. According to the ICAO recommendation, the P2 denotes "return no file information", and there is no Le byte present. Therefore, the response data field MUST be empty. The eMRTD MUST return status bytes ‘90 00’.
Postconditions	LDS application is selected.

3.1.2 Test Case 7816_A_2

Purpose	Selecting the LDS Application using the AID (robustness tests)
Version	2.04
References	ICAO LDS [R1], [R2]
Profile	ICAO, Plain
Preconditions	LDS application MUST NOT be selected.
Test scenario	1. Send the following SelectApplication APDU to the eMRTD. => ‘8F A4 04 0C 07 A0 00 00 02 47 10 01’ 2. Send the following SelectApplication APDU to the eMRTD. => ‘00 A4 04 0C 07 A0 00 00 02 47 10 02’ 3. Send the following SelectApplication APDU to the eMRTD. => ‘00 A4 84 0C 07 A0 00 00 02 47 10 01’ 4. Send the following SelectApplication APDU to the eMRTD. => ‘00 A4 04 8C 07 A0 00 00 02 47 10 01’ 5. Send the following SelectApplication APDU to the eMRTD. => ‘00 A4 04 0C 08 A0 00 00 02 47 10 01’ 6. Send the following SelectApplication APDU twice to the eMRTD. => ‘00 A4 04 0C 07 A0 00 00 02 47 10 01’ => ‘00 A4 04 0C 07 A0 00 00 02 47 10 01’
Expected results	1. The given APDU has an invalid class byte that is explicitly defined as invalid in ISO 7816-4 [R5]. Therefore, the eMRTD MUST return an ISO checking error or ISO execution error . 2. The APDU has an invalid AID that does not belong to LDS application. Therefore, the eMRTD MUST return an ISO checking error or ISO execution error . 3. The APDU has an invalid P1 parameter. Therefore, the eMRTD chip MUST return an ISO checking error or ISO execution error . 4. The APDU has an invalid P2 parameter. Therefore, the eMRTD chip MUST return an ISO checking error or ISO execution error . 5. The APDU has an invalid LC parameter. Therefore, the eMRTD chip MUST return an ISO checking error or ISO execution error . 6. The application MUST be selected successfully even it was already selected before. Therefore, the eMRTD MUST return the status bytes ‘90 00’ twice.
Postconditions	LDS application is selected.

3.2 Unit Test ISO_7816_B – Security conditions of BAC protected eMRTDs

This unit tests the security conditions of a BAC protected eMRTD. It MUST NOT be possible read the content of any present file. The tests of this unit try to access the files with an explicit SelectFile command, a ReadBinary command with implicit file selection via the short file identifier (SFI), and unsecured ReadBinary while access is granted. Note: Some eMRTDs allow selection of a protected file but no read access to this file, which complies with and the latest version of [R4].

The tests in this unit only apply to BAC protected eMRTDs (profile BAC).

The tests in this unit do not test the secure messaging implementation including postconditions (e.g. SM termination); therefore, status bytes MAY be returned in secure messaging or without it. Unit 7816_C handles this. In the following test cases, “basic access is refused” means that protected data cannot be accessed. The term “basic access is granted” means that the inspection system has successfully authenticated to the eMRTD.

3.2.1 Test Case 7816_B_1

Purpose	Accessing the EF.COM file with explicit file selection
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the following SelectApplication APDU to the eMRTD. => ‘00 A4 02 0C 02 01 1E’
Expected results	1. The eMRTD MUST return status bytes ‘69 82’ or ‘90 00’.
Postconditions	Preconditions remain unchanged.

3.2.2 Test Case 7816_B_2

Purpose	Accessing the EF.SOD file with explicit file selection
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the following SelectApplication APDU to the eMRTD. => ‘00 A4 02 0C 02 01 1D’
Expected results	1. The eMRTD MUST return status bytes ‘69 82’ or ‘90 00’.
Postconditions	Preconditions remain unchanged.

3.2.3 Test Case 7816_B_3

Purpose	Accessing the EF.DG1 file with explicit file selection
Version	1.1
References	ICAO LDS and PKI [R1], [R2] and PKI
Profile	BAC
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the following SelectApplication APDU to the eMRTD. => ‘00 A4 02 0C 02 01 01’
Expected results	1. The eMRTD MUST return status bytes ‘69 82’ or 90 00.
Postconditions	Preconditions remain unchanged.

3.2.4 Test Case 7816_B_4

Purpose	Accessing the EF.DG2 file with explicit file selection
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the following SelectApplication APDU to the eMRTD. => ‘00 A4 02 0C 02 01 02’
Expected results	1. The eMRTD MUST return status bytes ‘69 82’ or ‘90 00’.
Postconditions	Preconditions remain unchanged.

3.2.5 Test Case 7816_B_5

Purpose	Accessing the EF.DG3 file with explicit file selection
Version	2.02
References	ICAO LDS and PKI [R1], [R2]
Profile	(BAC, DG3) or (BAC, EAC, DG3)
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the following SelectApplication APDU to the eMRTD. => ‘00 A4 02 0C 02 01 03’
Expected results	1. The eMRTD MUST return status bytes ‘69 82’ or ‘90 00’.
Postconditions	Preconditions remain unchanged.

3.2.6 Test Case 7816_B_6

Purpose	Accessing the EF.DG4 file with explicit file selection
Version	2.02
References	ICAO LDS and PKI [R1], [R2]
Profile	(BAC, DG4) or (BAC, EAC, DG4)
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the following SelectApplication APDU to the eMRTD. => ‘00 A4 02 0C 02 01 04’
Expected results	1. The eMRTD MUST return status bytes ‘69 82’ or ‘90 00’.
Postconditions	Preconditions remain unchanged.

3.2.7 Test Case 7816_B_7

Purpose	Accessing the EF.DG5 file with explicit file selection
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC, DG5
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the following SelectApplication APDU to the eMRTD. => ‘00 A4 02 0C 02 01 05’
Expected results	1. The eMRTD MUST return status bytes ‘69 82’ or ‘90 00’.
Postconditions	Preconditions remain unchanged.

3.2.8 Test Case 7816_B_8

Purpose	Accessing the EF.DG6 file with explicit file selection
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC, DG6
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the following SelectApplication APDU to the eMRTD. => ‘00 A4 02 0C 02 01 06’
Expected results	1. The eMRTD MUST return status bytes ‘69 82’ or ‘90 00’.
Postconditions	Preconditions remain unchanged.

3.2.9 Test Case 7816_B_9

Purpose	Accessing the EF.DG7 file with explicit file selection
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC, DG7
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the following SelectApplication APDU to the eMRTD. => ‘00 A4 02 0C 02 01 07’
Expected results	1. The eMRTD MUST return status bytes ‘69 82’ or ‘90 00’.
Postconditions	Preconditions remain unchanged.

3.2.10 Test Case 7816_B_10

Purpose	Accessing the EF.DG8 file with explicit file selection
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC, DG8
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the following SelectApplication APDU to the eMRTD. => ‘00 A4 02 0C 02 01 08’
Expected results	1. The eMRTD MUST return status bytes ‘69 82’ or ‘90 00’.
Postconditions	Preconditions remain unchanged.

3.2.11 Test Case 7816_B_11

Purpose	Accessing the EF.DG9 file with explicit file selection
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC, DG9
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the following SelectApplication APDU to the eMRTD. => ‘00 A4 02 0C 02 01 09’
Expected results	1. The eMRTD MUST return status bytes ‘69 82’ or ‘90 00’.
Postconditions	Preconditions remain unchanged.

3.2.12 Test Case 7816_B_12

Purpose	Accessing the EF.DG10 file with explicit file selection
Version	1.1
References	ICAO LDS and PKI [R1], [R2]]
Profile	BAC, DG10
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the following SelectApplication APDU to the eMRTD. => '00 A4 02 0C 02 01 0A'
Expected results	1. The eMRTD MUST return status bytes '69 82' or '90 00'.
Postconditions	Preconditions remain unchanged.

3.2.13 Test Case 7816_B_13

Purpose	Accessing the EF.DG11 file with explicit file selection
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC, DG11
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the following SelectApplication APDU to the eMRTD. => '00 A4 02 0C 02 01 0B'
Expected results	1. The eMRTD MUST return status bytes '69 82' or '90 00'.
Postconditions	Preconditions remain unchanged.

3.2.14 Test Case 7816_B_14

Purpose	Accessing the EF.DG12 file with explicit file selection
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC, DG12
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the following SelectApplication APDU to the eMRTD. => '00 A4 02 0C 02 01 0C'
Expected results	1. The eMRTD MUST return status bytes '69 82' or '90 00'.
Postconditions	Preconditions remain unchanged.

3.2.15 Test Case 7816_B_15

Purpose	Accessing the EF.DG13 file with explicit file selection
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC, DG13
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the following SelectApplication APDU to the eMRTD. => ‘00 A4 02 0C 02 01 0D’
Expected results	1. The eMRTD MUST return status bytes ‘69 82’ or ‘90 00’.
Postconditions	Preconditions remain unchanged.

3.2.16 Test Case 7816_B_16

Purpose	Accessing the EF.DG14 file with explicit file selection
Version	2.02
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC, (EAC or PACE or AA-ECDSA)
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the following SelectApplication APDU to the eMRTD. => ‘00 A4 02 0C 02 01 0E’
Expected results	1. The eMRTD MUST return status bytes ‘69 82’ or ‘90 00’.
Postconditions	Preconditions remain unchanged.

3.2.17 Test Case 7816_B_17

Purpose	Accessing the EF.DG15 file with explicit file selection
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC, AA
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the following SelectApplication APDU to the eMRTD. => ‘00 A4 02 0C 02 01 0F’
Expected results	1. The eMRTD MUST return status bytes ‘69 82’ or ‘90 00’.
Postconditions	Preconditions remain unchanged.

3.2.18 Test Case 7816_B_18

Purpose	Accessing the EF.DG16 file with explicit file selection
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC, DG16
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the following SelectApplication APDU to the eMRTD. => ‘00 A4 02 0C 02 01 10’
Expected results	1. The eMRTD MUST return status bytes ‘69 82’ or ‘90 00’.
Postconditions	Preconditions remain unchanged.

3.2.19 Test Case 7816_B_19

Purpose	Accessing the EF.COM file with implicit file selection (ReadBinary with SFI)
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD. => ‘00 B0 9E 00 00’
Expected results	1. Since read access is prohibited without BAC, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’.
Postconditions	Preconditions remain unchanged.

3.2.20 Test Case 7816_B_20

Purpose	Accessing the EF.SOD file with implicit file selection (ReadBinary with SFI)
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD. => ‘00 B0 9D 00 00’
Expected results	1. Since read access is prohibited without BAC, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’.
Postconditions	Preconditions remain unchanged.

3.2.21 Test Case 7816_B_21

Purpose	Accessing the EF.DG1 file with implicit file selection (ReadBinary with SFI)
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD. => ‘00 B0 81 00 00’
Expected results	1. Since read access is prohibited without BAC, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’.
Postconditions	Preconditions remain unchanged.

3.2.22 Test Case 7816_B_22

Purpose	Accessing the EF.DG2 file with implicit file selection (ReadBinary with SFI)
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD. => ‘00 B0 82 00 00’
Expected results	1. Since read access is prohibited without BAC, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’.
Postconditions	Preconditions remain unchanged.

3.2.23 Test Case 7816_B_23

Purpose	Accessing the EF.DG3 file with implicit file selection (ReadBinary with SFI)
Version	2.02
References	ICAO LDS and PKI [R1], [R2]
Profile	(BAC, DG3) or (BAC, EAC, DG3)
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD. => ‘00 B0 83 00 00’
Expected results	1. Since read access is prohibited without BAC/EAC, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’.
Postconditions	Preconditions remain unchanged.

3.2.24 Test Case 7816_B_24

Purpose	Accessing the EF.DG4 file with implicit file selection (ReadBinary with SFI)
Version	2.02
References	ICAO LDS and PKI [R1], [R2]
Profile	(BAC, DG4) or (BAC, EAC, DG4)
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD. => ‘00 B0 84 00 00’
Expected results	1. Since read access is prohibited without BAC/EAC, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’.
Postconditions	Preconditions remain unchanged.

3.2.25 Test Case 7816_B_25

Purpose	Accessing the EF.DG5 file with implicit file selection (ReadBinary with SFI)
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC, DG5
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD. => ‘00 B0 85 00 00’
Expected results	1. Since read access is prohibited without BAC, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’.
Postconditions	Preconditions remain unchanged.

3.2.26 Test Case 7816_B_26

Purpose	Accessing the EF.DG6 file with implicit file selection (ReadBinary with SFI)
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC, DG6
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD. => ‘00 B0 86 00 00’
Expected results	1. Since read access is prohibited without BAC, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’.
Postconditions	Preconditions remain unchanged.

3.2.27 Test Case 7816_B_27

Purpose	Accessing the EF.DG7 file with implicit file selection (ReadBinary with SFI)
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC, DG7
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD. => ‘00 B0 87 00 00’
Expected results	1. Since read access is prohibited without BAC, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’.
Postconditions	Preconditions remain unchanged.

3.2.28 Test Case 7816_B_28

Purpose	Accessing the EF.DG8 file with implicit file selection (ReadBinary with SFI)
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC, DG8
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD. => ‘00 B0 88 00 00’
Expected results	1. Since read access is prohibited without BAC, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’.
Postconditions	Preconditions remain unchanged.

3.2.29 Test Case 7816_B_29

Purpose	Accessing the EF.DG9 file with implicit file selection (ReadBinary with SFI)
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC, DG9
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD. => ‘00 B0 89 00 00’
Expected results	1. Since read access is prohibited without BAC, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’.
Postconditions	Preconditions remain unchanged.

3.2.30 Test Case 7816_B_30

Purpose	Accessing the EF.DG10 file with implicit file selection (ReadBinary with SFI)
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC, DG10
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD. => ‘00 B0 8A 00 00’
Expected results	1. Since read access is prohibited without BAC, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’.
Postconditions	Preconditions remain unchanged.

3.2.31 Test Case 7816_B_31

Purpose	Accessing the EF.DG11 file with implicit file selection (ReadBinary with SFI)
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC, DG11
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD. => ‘00 B0 8B 00 00’
Expected results	1. Since read access is prohibited without BAC, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’.
Postconditions	Preconditions remain unchanged.

3.2.32 Test Case 7816_B_32

Purpose	Accessing the EF.DG12 file with implicit file selection (ReadBinary with SFI)
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC, DG12
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD. => ‘00 B0 8C 00 00’
Expected results	1. Since read access is prohibited without BAC, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’.
Postconditions	Preconditions remain unchanged.

3.2.33 Test Case 7816_B_33

Purpose	Accessing the EF.DG13 file with implicit file selection (ReadBinary with SFI)
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC, DG13
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD. => ‘00 B0 8D 00 00’
Expected results	1. Since read access is prohibited without BAC, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’.
Postconditions	Preconditions remain unchanged.

3.2.34 Test Case 7816_B_34

Purpose	Accessing the EF.DG14 file with implicit file selection (ReadBinary with SFI)
Version	2.02
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC, (EAC or PACE or AA-ECDSA)
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD. => ‘00 B0 8E 00 00’
Expected results	1. Since read access is prohibited without BAC or PACE, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’.
Postconditions	Preconditions remain unchanged.

3.2.35 Test Case 7816_B_35

Purpose	Accessing the EF.DG15 file with implicit file selection (ReadBinary with SFI)
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC, AA
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD. => ‘00 B0 8F 00 00’
Expected results	1. Since read access is prohibited without BAC, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’.
Postconditions	Preconditions remain unchanged.

3.2.36 Test Case 7816_B_36

Purpose	Accessing the EF.DG16 file with implicit file selection (ReadBinary with SFI)
Version	1.1
References	ICAO LDS and PKI [R1], [R2]
Profile	BAC, DG16
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD. => ‘00 B0 90 00 00’
Expected results	1. Since read access is prohibited without BAC, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’.
Postconditions	Preconditions remain unchanged.

3.2.37 Test Case 7816_B_37

Purpose	Accessing the EF.COM file with ReadBinary. The test verifies the enforcement of Secure Messaging while basic access is granted.
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.COM encoded as a valid SM APDU to the eMRTD. => ‘0C B0 9E 00 0D 97 01 06 8E 08 <checksum> 00’ 2. Send the following ReadBinary APDU as a plain unprotected APDU to the eMRTD. => ‘00 B0 00 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.2.38 Test Case 7816_B_38

Purpose	Accessing the EF. SOD file with ReadBinary. The test verifies the enforcement of Secure Messaging while basic access is granted.
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.SOD encoded as a valid SM APDU to the eMRTD. => ‘0C B0 9D 00 0D 97 01 06 8E 08 <checksum> 00’ 2. Send the following ReadBinary APDU as a plain unprotected APDU to the eMRTD. => ‘00 B0 00 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.2.39 Test Case 7816_B_39

Purpose	Accessing the EF. DG1 file with ReadBinary. The test verifies the enforcement of Secure Messaging while basic access is granted.
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.DG1 encoded as a valid SM APDU to the eMRTD. => ‘0C B0 81 00 0D 97 01 06 8E 08 <checksum> 00’ 2. Send the following ReadBinary APDU as a plain unprotected APDU to the eMRTD. => ‘00 B0 00 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.2.40 Test Case 7816_B_40

Purpose	Accessing the EF. DG2 file with ReadBinary. The test verifies the enforcement of Secure Messaging while basic access is granted.
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.DG2 encoded as a valid SM APDU to the eMRTD. => ‘0C B0 82 00 0D 97 01 06 8E 08 <checksum> 00’ 2. Send the following ReadBinary APDU as a plain unprotected APDU to the eMRTD. => ‘00 B0 00 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.2.41 Test Case 7816_B_41

Purpose	Accessing the EF. DG3 file with ReadBinary. The test verifies the enforcement of Secure Messaging while basic or extended access is granted.
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	(BAC, DG3) or (BAC, EAC, DG3)
Preconditions	The LDS application MUST be selected and basic (extended) access MUST be granted.
Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.DG3 encoded as a valid SM APDU to the eMRTD. => ‘0C B0 83 00 0D 97 01 06 8E 08 <checksum> 00’ 2. Send the following ReadBinary APDU as a plain unprotected APDU to the eMRTD. => ‘00 B0 00 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.2.42 Test Case 7816_B_42

Purpose	Accessing the EF. DG4 file with ReadBinary. The test verifies the enforcement of Secure Messaging while basic or extended access is granted.
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	(BAC, DG4) or (BAC, EAC, DG4)
Preconditions	The LDS application MUST be selected and basic (extended) access MUST be granted.
Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.DG4 encoded as a valid SM APDU to the eMRTD. => ‘0C B0 84 00 0D 97 01 06 8E 08 <checksum> 00’ 2. Send the following ReadBinary APDU as a plain unprotected APDU to the eMRTD. => ‘00 B0 00 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.2.43 Test Case 7816_B_43

Purpose	Accessing the EF. DG5 file with ReadBinary. The test verifies the enforcement of Secure Messaging while basic access is granted.
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC, DG5
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.DG5 encoded as a valid SM APDU to the eMRTD. => ‘0C B0 85 00 0D 97 01 06 8E 08 <checksum> 00’ 2. Send the following ReadBinary APDU as a plain unprotected APDU to the eMRTD. => ‘00 B0 00 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.2.44 Test Case 7816_B_44

Purpose	Accessing the EF. DG6 file with ReadBinary. The test verifies the enforcement of Secure Messaging while basic access is granted.
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC, DG6
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.DG6 encoded as a valid SM APDU to the eMRTD. => ‘0C B0 86 00 0D 97 01 06 8E 08 <checksum> 00’ 2. Send the following ReadBinary APDU as a plain unprotected APDU to the eMRTD. => ‘00 B0 00 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.2.45 Test Case 7816_B_45

Purpose	Accessing the EF. DG7 file with ReadBinary. The test verifies the enforcement of Secure Messaging while basic access is granted.
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC, DG7
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.DG7 encoded as a valid SM APDU to the eMRTD. => '0C B0 87 00 0D 97 01 06 8E 08 <checksum> 00' 2. Send the following ReadBinary APDU as a plain unprotected APDU to the eMRTD. => '00 B0 00 00 00'
Expected results	1. The eMRTD MUST return the status bytes '90 00'. 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.2.46 Test Case 7816_B_46

Purpose	Accessing the EF. DG8 file with ReadBinary. The test verifies the enforcement of Secure Messaging while basic access is granted.
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC, DG8
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.DG8 encoded as a valid SM APDU to the eMRTD. => '0C B0 88 00 0D 97 01 06 8E 08 <checksum> 00' 2. Send the following ReadBinary APDU as a plain unprotected APDU to the eMRTD. => '00 B0 00 00 00'
Expected results	1. The eMRTD MUST return the status bytes '90 00'. 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.2.47 Test Case 7816_B_47

Purpose	Accessing the EF. DG9 file with ReadBinary. The test verifies the enforcement of Secure Messaging while basic access is granted.
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC, DG9
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.DG9 encoded as a valid SM APDU to the eMRTD. => ‘0C B0 89 00 0D 97 01 06 8E 08 <checksum> 00’ 2. Send the following ReadBinary APDU as a plain unprotected APDU to the eMRTD. => ‘00 B0 00 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.2.48 Test Case 7816_B_48

Purpose	Accessing the EF. DG10 file with ReadBinary. The test verifies the enforcement of Secure Messaging while basic access is granted.
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC, DG10
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.DG10 encoded as a valid SM APDU to the eMRTD. => ‘0C B0 8A 00 0D 97 01 06 8E 08 <checksum> 00’ 2. Send the following ReadBinary APDU as a plain unprotected APDU to the eMRTD. => ‘00 B0 00 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.2.49 Test Case 7816_B_49

Purpose	Accessing the EF. DG11 file with ReadBinary. The test verifies the enforcement of Secure Messaging while basic access is granted.
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC, DG11
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.DG11 encoded as a valid SM APDU to the eMRTD. => ‘0C B0 8B 00 0D 97 01 06 8E 08 <checksum> 00’ 2. Send the following ReadBinary APDU as a plain unprotected APDU to the eMRTD. => ‘00 B0 00 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.2.50 Test Case 7816_B_50

Purpose	Accessing the EF. DG12 file with ReadBinary. The test verifies the enforcement of Secure Messaging while basic access is granted.
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC, DG12
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.DG12 encoded as a valid SM APDU to the eMRTD. => ‘0C B0 8C 00 0D 97 01 06 8E 08 <checksum> 00’ 2. Send the following ReadBinary APDU as a plain unprotected APDU to the eMRTD. => ‘00 B0 00 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.2.51 Test Case 7816_B_51

Purpose	Accessing the EF. DG13 file with ReadBinary. The test verifies the enforcement of Secure Messaging while basic access is granted.
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC, DG13
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.DG13 encoded as a valid SM APDU to the eMRTD. => ‘0C B0 8D 00 0D 97 01 06 8E 08 <checksum> 00’ 2. Send the following ReadBinary APDU as a plain unprotected APDU to the eMRTD. => ‘00 B0 00 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.2.52 Test Case 7816_B_52

Purpose	Accessing the EF. DG14 file with ReadBinary. The test verifies the enforcement of Secure Messaging while basic access is granted.
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC, (EAC or PACE or AA-ECDSA)
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.DG14 encoded as a valid SM APDU to the eMRTD. => ‘0C B0 8E 00 0D 97 01 06 8E 08 <checksum> 00’ 2. Send the following ReadBinary APDU as a plain unprotected APDU to the eMRTD. => ‘00 B0 00 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.2.53 Test Case 7816_B_53

Purpose	Accessing the EF. DG15 file with ReadBinary. The test verifies the enforcement of Secure Messaging while basic access is granted.
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC, AA
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.DG15 encoded as a valid SM APDU to the eMRTD. => ‘0C B0 8F 00 0D 97 01 06 8E 08 <checksum> 00’ 2. Send the following ReadBinary APDU as a plain unprotected APDU to the eMRTD. => ‘00 B0 00 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.2.54 Test Case 7816_B_54

Purpose	Accessing the EF. DG16 file with ReadBinary. The test verifies the enforcement of Secure Messaging while basic access is granted.
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC, DG16
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.DG16 encoded as a valid SM APDU to the eMRTD. => ‘0C B0 ‘90 00’ 0D 97 01 06 8E 08 <checksum> 00’ 2. Send the following ReadBinary APDU as a plain unprotected APDU to the eMRTD. => ‘00 B0 00 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.3 Unit Test ISO_7816_C – Basic Access Control

This unit checks the BAC implementation of the eMRTD. The complete BAC access mechanism is tested, including robustness tests with invalid input data.

Since the tests in this unit apply to BAC protected eMRTDs, they are only mandatory for eMRTDs complying with the BAC profile.

In the following test cases, “basic access is refused” means that there are no valid session keys for secure messaging available and that access to any BAC protected file is refused. The term “basic access is granted” means that the inspection system has successfully authenticated to the eMRTD and that valid session keys are available for secure messaging.

The READ BINARY command in SM mode is used in the following test cases to verify that the session keys are no longer valid. Alternatively, the command SELECT FILE in SM mode MAY be used.

3.3.1 Test Case 7816_C_1

Purpose	This function verifies the GetChallenge command (positive test).
Version	1.1
References	ICAO PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected and basic access MUST be refused.
Test scenario	1. Send the following GetChallenge APDU to the eMRTD. => ‘00 84 00 00 08’ 2. Send the same GetChallenge APDU to the eMRTD. => ‘00 84 00 00 08’
Expected results	1. The eMRTD MUST return 8 random bytes and the status bytes ‘90 00’. 2. The eMRTD MUST return 8 different random bytes and the status bytes ‘90 00’.
Postconditions	Preconditions remain unchanged.

3.3.2 Test Case 7816_C_2

Purpose	This test checks the response to the MutualAuthenticate command (positive test).
Version	1.1
References	ICAO PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected and basic access MUST be refused.
Test scenario	1. Send the following GetChallenge APDU to the eMRTD. => ‘00 84 00 00 08’ 2. Send the MutualAuthenticate APDU to the eMRTD. The <data> MUST be calculated from the given MRZ data and the challenge returned in step 1. => ‘00 82 00 00 28 <data> 28’
Expected results	1. The eMRTD MUST return 8 random bytes and the status bytes ‘90 00’. 2. The response from the eMRTD MUST be verified as specified in [R2]. The returned status bytes MUST be ‘90 00’.
Postconditions	Basic access is granted.

3.3.3 Test Case 7816_C_3

Purpose	This test checks the authentication failure response to the MutualAuthenticate command
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected and basic access MUST be refused.
Test scenario	1. Send the following GetChallenge APDU to the eMRTD. => ‘00 84 00 00 08’ 2. Send the MutualAuthenticate APDU to the eMRTD. Same as 7816_C_2, but for the <data> calculation data from a different MRZ MUST be used. To achieve this, the document number MUST be increment by 1 before the <data> is calculated. => ‘00 82 00 00 28 <data> 28’
Expected results	1. The eMRTD MUST return 8 random bytes and the status bytes ‘90 00’. 2. The eMRTD MUST respond with an ISO warning or ISO checking error or ISO execution error .
Postconditions	Preconditions remain unchanged.

3.3.4 Test Case 7816_C_4

Purpose	This test checks the authentication failure response to the MutualAuthenticate command
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected and basic access MUST be refused. The GetChallenge command MUST NOT have been executed.
Test scenario	1. Send the MutualAuthenticate APDU to the eMRTD. Same as 7816_C_2, but for the <data> calculation the challenge ’00 00 00 00 00 00 00 00’ MUST be used. => ‘00 82 00 00 28 <data> 28’ 2. Send the following GetChallenge APDU to the eMRTD. => ‘00 84 00 00 08’ 3. Send the following GetChallenge APDU to the eMRTD. => ‘00 84 00 00 08’ 4. Send the MutualAuthenticate APDU to the eMRTD. Same as 7816_C_2, but for the <data> calculation the challenge of step 2 MUST be used. => ‘00 82 00 00 28 <data> 28’
Expected results	1. The eMRTD MUST respond with an ISO warning or ISO checking error or ISO execution error . 2. The eMRTD MUST return 8 random bytes and the status bytes ‘90 00’. 3. The eMRTD MUST return 8 random bytes and the status bytes ‘90 00’. 4. The eMRTD MUST respond with an ISO warning or ISO checking error or ISO execution error .
Postconditions	Preconditions remain unchanged.

3.3.5 Test Case 7816_C_5

Purpose	This test checks the response for the MutualAuthenticate command (robustness test)
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected and basic access MUST be refused.
Test scenario	1. Send the following GetChallenge APDU to the eMRTD. => ‘00 84 00 00 08’ 2. Send the MutualAuthenticate APDU to the eMRTD. The <data> MUST be calculated from the given MRZ data and the challenge returned in step 1. The class byte is set to a wrong value. => ‘8F 82 00 00 28 <data> 28’ 3. Send the following GetChallenge APDU to the eMRTD. => ‘00 84 00 00 08’ 4. Send the MutualAuthenticate APDU to the eMRTD. The <data> MUST be calculated from the given MRZ data and the challenge returned in step 3. The P1 byte is set to a wrong value. => ‘00 82 60 00 28 <data> 28’ 5. Send the following GetChallenge APDU to the eMRTD. => ‘00 84 00 00 08’ 6. Send the MutualAuthenticate APDU to the eMRTD. The <data> MUST be calculated from the given MRZ data and the challenge returned in step 5. The P2 byte is set to a wrong value. => ‘00 82 00 60 28 <data> 28’ 7. Send the following GetChallenge APDU to the eMRTD. => ‘00 84 00 00 08’ 8. Send the MutualAuthenticate APDU to the eMRTD. The <data> MUST be calculated from the given MRZ data and the challenge returned in step 7. The LC byte is set to a wrong value. => ‘00 82 00 00 29 <data> 28’
Expected results	1. The eMRTD MUST return 8 random bytes and a ‘90 00’ status byte. 2. The eMRTD MUST respond with an ISO checking error or ISO execution error . 3. The eMRTD MUST return 8 random bytes and the status bytes ‘90 00’. 4. The eMRTD MUST respond with an ISO checking error or ISO execution error . 5. The eMRTD MUST return 8 random bytes and a ‘90 00’ status byte. 6. The eMRTD MUST respond with an ISO checking error or ISO execution error . 7. The eMRTD MUST return 8 random bytes and the status bytes ‘90 00’. 8. The eMRTD MUST respond with an ISO checking error or ISO execution error .
Postconditions	Preconditions remain unchanged.

3.3.6 Test Case 7816_C_6

Purpose	This test checks the response for the MutualAuthenticate command with a corrupted MAC.
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected and basic access MUST be refused.
Test scenario	1. Send the following GetChallenge APDU to the eMRTD. => ‘00 84 00 00 08’ 2. Send the MutualAuthenticate APDU to the eMRTD. The <data> MUST be calculated from the given MRZ data and the challenge returned in step 1. In the calculated MAC the very last byte is incremented by one. => ‘00 82 00 00 28 <data> 28’
Expected results	1. The eMRTD MUST return 8 random bytes and the status bytes ‘90 00’. 2. The eMRTD MUST respond with an ISO warning or ISO checking error or ISO execution error .
Postconditions	Preconditions remain unchanged.

Note: this test case differs from test case ISO17816_C_3. In this test case, only the MAC is manipulated but the cryptogram is valid.

3.3.7 Test Case 7816_C_7

Test case deleted because the GetChallenge command using secure messaging is not defined. The test case may be added again when the EAC specification is finalized.

3.3.8 Test Case 7816_C_8

Purpose	This test checks the Secure Messaging coding of a ReadBinary (B0) with SFI (positive tests)
Version	1.1
References	ICAO PKI [R1], [R2] Supplement S3.0-20050531-PKI0033 [R4].
Profile	BAC
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. Send the following ReadBinary (SFI) APDU encoded as a valid SM APDU to the eMRTD. => ‘0C B0 9E 00 0D 97 01 06 8E 08 <checksum> 00’ 2. Search for the cryptogram DO encoded in tag ‘87’ and decrypt it with current session key. 3. Search for the processing status DO encoded in tag ‘99’ and verify status bytes received. 4. Search for the cryptographic checksum DO encoded in tag ‘8E’ and verify it with current session key.
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The response of step 1 MUST contain the read data in a valid cryptogram encoded in tag ‘87’. 3. The response of step 1 SHOULD contain SW1-SW2 encoded in tag ‘99’ that equals the status bytes of the secured response. 4. The response of step 1 MUST contain a valid cryptographic checksum encoded in tag ‘8E’.
Postconditions	Preconditions remain unchanged.

3.3.9 Test Case 7816_C_9

Purpose	This test checks the Secure Messaging coding of a ReadBinary (‘B1’) with SFI (positive tests)
Version	1.1
References	ICAO PKI [R1], [R2] ISO 7816-4 for TLV encoded data objects [R5] Supplement S3.0-20050531-PKI0033 [R4]
Profile	BAC, OddIns
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. Send the following ReadBinary (SFI) APDU encoded as a valid SM APDU to the eMRTD. The offset (0) MUST be encoded in a DO ‘54’[1], which is then encrypted in a SM ‘85’ object. => ‘0C B1 00 1E 17 85 08 <cryptogram> 97 01 06 8E 08 <checksum> 00’ 2. Search for the cryptogram DO encoded in tag ‘85’ and decrypt it with current session key. 3. Search for the processing status DO encoded in tag ‘99’ and verify status bytes received. 4. Search for the cryptographic checksum DO encoded in tag ‘8E’ and verify it with current session key.
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The response of step 1 MUST contain the read data in a valid cryptogram encoded in tag ‘85’. The data MUST be encapsulated in a tag ‘53’ object. 3. The response of step 1 SHOULD contain SW1-SW2 encoded in tag ‘99’ that equals the status bytes of the secured response. 4. The response of step 1 MUST contain a valid cryptographic checksum encoded in tag ‘8E’.
Postconditions	Preconditions remain unchanged.

[1] It is RECOMMENDED to encode the offset in tag 54 in one byte, i.e. ’54 01 00’. This issue requires further clarification and will be forwarded to TF1/TF5.

3.3.10 Test Case 7816_C_10

Purpose	This test checks the Secure Messaging coding of a SelectFile and ReadBinary (B0) w/o SFI (positive tests)
Version	1.1
References	ICAO PKI [R1], [R2] Supplement S3.0-20050531-PKI0033 [R4]
Profile	BAC
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. EF.COM SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 1E’. Send the following SelectFile APDU encoded as a valid SM APDU to the eMRTD. => ‘0C A4 02 0C 15 87 09 01 <cryptogram> 8E 08 <checksum> 00’ 2. Search for the processing status DO encoded in tag ‘99’ and verify status bytes received. 3. Search for the cryptographic checksum DO encoded in tag ‘8E’ and verify it with current session key. 4. Send the following ReadBinary APDU encoded as a valid SM APDU to the eMRTD. => ‘0C B0 00 00 0D 97 01 06 8E 08 <checksum> 00’ 5. Search for the cryptogram DO encoded in tag ‘87’ and decrypt it with current session key. 6. Search for the processing status DO encoded in tag ‘99’ and verify status bytes received. 7. Search for the cryptographic checksum DO encoded in tag ‘8E’ and verify it with current session key. 8. Search for further DO.
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The response of step 1 MUST contain SW1-SW2 encoded in tag ‘99’ that MUST equal the received status bytes of the secured response. 3. The response of step 1 MUST contain a valid cryptographic checksum encoded in tag ‘8E’. 4. The eMRTD MUST return the status bytes ‘90 00’. 5. The response of step 4 MUST contain the read data in a valid cryptogram encoded in tag ‘87’. 6. The response of step 4 SHOULD contain SW1-SW2 encoded in tag ‘99’ that equals the received status bytes of the secured response. 7. The response of step 4 MUST contain a valid cryptographic checksum encoded in tag ‘8E’. 8. The response MUST NOT contain any further data but the response trailer.
Postconditions	Preconditions remain unchanged.

3.3.11 Test Case 7816_C_11

Purpose	This test checks the Secure Messaging coding of a SelectFile and ReadBinary (B1) w/o SFI (positive tests)
Version	1.1
References	ICAO PKI [R1], [R2] ISO 7816-4 [R5] for TLV encoded data objects Supplement S3.0-20050531-PKI0033 [R4]
Profile	BAC, OddIns
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. EF.COM SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 1E’. Send the following SelectFile APDU encoded as a valid SM APDU to the eMRTD. => ‘0C A4 02 0C 15 87 09 01 <cryptogram> 8E 08 <checksum> 00’ 2. Search for the processing status DO encoded in tag ‘99’ and verify status bytes received. 3. Search for the cryptographic checksum DO encoded in tag 8E and verify it with current session key. 4. Send the following ReadBinary APDU encoded as a valid SM APDU to the eMRTD. The offset (0) MUST be encoded in a DO ‘54’[2], which is then encrypted in a SM ‘85’ object. => ‘0C B1 00 00 17 85 08 <cryptogram> 97 01 06 8E 08 <checksum> 00’ 5. Search for the cryptogram DO encoded in tag ‘85’ and decrypt it with current session key. 6. Search for the processing status DO encoded in tag ‘99’ and verify status bytes received. 7. Search for the cryptographic checksum DO encoded in tag ‘8E’ and verify it with current session key. 8. Search for further DO.
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The response of step 1 MUST contain SW1-SW2 encoded in tag ‘99’ that MUST equal the received status bytes of the secured response. 3. The response of step 1 MUST contain a valid cryptographic checksum encoded in tag ‘8E’. 4. The eMRTD MUST return the status bytes ‘90 00’. 5. The response of step 4 MUST contain the read data in a valid cryptogram encoded in tag ‘85’. The data MUST be encapsulated in a tag ‘53’ object. 6. The response of step 4 SHOULD contain SW1-SW2 encoded in tag ‘99’ that equals the received status bytes of the secured response. 7. The response of step 4 MUST contain a valid cryptographic checksum encoded in tag ‘8E’. 8. The response MUST NOT contain any further data but the response trailer.
Postconditions	Preconditions remain unchanged.

[2] It is RECOMMENDED to encode the offset in tag 54 in one byte, i.e. ’54 01 00’. This issue requires further clarification and will be forwarded to TF1/TF5.

3.3.12 Test Case 7816_C_12

Purpose	The test verifies the Secure Messaging handling while basic access is granted for the SelectFile Command (checksum missing)
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. EF.COM SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 1E’. Send the following SelectFile APDU encoded as a SM APDU but without the checksum SM object to the eMRTD. => ‘0C A4 02 0C 0B 87 09 01 <cryptogram> 00’ 2. To verify that the error in step 1 has terminated the SM session, send a valid SM APDU (ReadBinary) to the eMRTD. => ‘0C B0 9E 00 0D 97 01 06 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error . 2. Since the session keys are no longer valid, the eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.3.13 Test Case 7816_C_13

Purpose	The test verifies the Secure Messaging handling while basic access is granted for the SelectFile Command (checksum corrupted)
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. EF.COM SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 1E’. Send the following SelectFile APDU encoded as a valid SM APDU to the eMRTD. The last byte of the checksum is incremented by one. => ‘0C A4 02 0C 15 87 09 01 <cryptogram> 8E 08 <corrupted checksum> 00’ 2. To verify that the error in step 1 has terminated the SM session, send a valid SM APDU (ReadBinary) to the eMRTD. => ‘0C B0 9E 00 0D 97 01 06 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return status bytes ‘69 88’ or ‘69 82’. 2. Since the session keys are no longer valid, the eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.3.14 Test Case 7816_C_14

Purpose	The tests verifies the Secure Messaging handling while basic access is granted for the SelectFile Command (bad send sequence counter)
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. EF.COM SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 1E’. Send the following SelectFile APDU encoded as a valid SM APDU to the eMRTD. During the coding of the SM APDU the SendSequenceCounter is not incremented. => ‘0C A4 02 0C 15 87 09 01 <cryptogram> 8E 08 <corrupted checksum> 00’ 2. To verify that the error in step 1 has terminated the SM session, send a valid SM APDU (ReadBinary) to the eMRTD. => ‘0C B0 9E 00 0D 97 01 06 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return status bytes ‘69 88’ or ‘69 82’. 2. Since the session keys are no longer valid, the eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.3.15 Test Case 7816_C_15

Purpose	The test verifies the Secure Messaging handling while basic access is granted for the SelectFile Command (invalid class byte)
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. EF.COM SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 1E’. Send the following SelectFile APDU encoded as a SM APDU to the eMRTD. The class byte is set to ‘00’. => ‘00 A4 02 0C 15 87 09 01 <cryptogram> 8E 08 <checksum> 00’ 2. To verify that the error in step 1 has terminated the SM session, send a valid SM APDU (ReadBinary) to the eMRTD. => ‘0C B0 9E 00 0D 97 01 06 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error . 2. Since the session keys are no longer valid, the eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.3.16 Test Case 7816_C_16

Purpose	The test verifies the enforcement of Secure Messaging while basic access is granted for the SelectFile Command.
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. EF.COM SHALL be selected. Send the following SelectFile APDU as a plain unprotected APDU to the eMRTD. => ‘00 A4 02 0C 02 01 1E’
Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error or status bytes ‘90 00’.
Postconditions	Postcondition depends on the status bytes returned by the eMRTD.

3.3.17 Test Case 7816_C_17

Purpose	The test verifies the Secure Messaging handling while basic access is granted for the ReadBinary Command (checksum missing).
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. Send the following ReadBinary APDU encoded as a SM APDU but without the checksum SM object to the eMRTD. => ‘0C B0 9E 00 03 97 01 06 00’ 2. To verify that the error in step 1 has terminated the SM session, send a valid SM APDU (ReadBinary) to the eMRTD. => ‘0C B0 9E 00 0D 97 01 06 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error . 2. Since the session keys are no longer valid, the eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.3.18 Test Case 7816_C_18

Purpose	The test verifies the Secure Messaging handling while basic access is granted for the ReadBinary Command (checksum corrupted).
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. Send the following ReadBinary APDU encoded as a valid SM APDU to the eMRTD. The last byte of the checksum is incremented by one. => ‘0C B0 00 00 0D 97 01 06 8E 08 <checksum> 00’ 2. To verify that the error in step 1 has terminated the SM session, send a valid SM APDU (ReadBinary) to the eMRTD. => ‘0C B0 9E 00 0D 97 01 06 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return status bytes ‘69 88’ or ‘69 82’. 2. Since the session keys are no longer valid, the eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.3.19 Test Case 7816_C_19

Purpose	The test verifies the Secure Messaging handling while basic access is granted for the ReadBinary Command (invalid class byte).
Version	2.04
References	ICAO PKI [R1], [R2]
Profile	BAC
Preconditions	The LDS application MUST be selected and basic access MUST be granted.
Test scenario	1. Send the following ReadBinary APDU encoded as a SM APDU to the eMRTD. The class byte is set to ‘00’. => ‘00 B0 00 00 0D 97 01 06 8E 08 <checksum> 00’ 2. To verify that the error in step 1 has terminated the SM session, send a valid SM APDU (ReadBinary) to the eMRTD. => ‘0C B0 9E 00 0D 97 01 06 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error . 2. Since the session keys are no longer valid, the eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Basic access is refused.

3.4 Unit Test ISO_7816_D – Protected SelectFile Command

This unit verifies the implementation of the protected SelectFile command.

The eMRTD MUST be BAC or/and PACE protected. For all test cases of unit test ISO_7816_D, basic access MUST be granted as tested in 7816_C_2 for BAC and 7816_P_1 for PACE. All APDUs MUST be correctly encoded for Secure Messaging and the eMRTD response MUST be correctly decoded again. The expected results of the test cases are plain text data after decoding the protected response APDU.

The tests in this unit do not test the secure messaging implementation including postconditions (e.g. SM termination); therefore, status bytes MAY be returned in secure messaging or without it. Unit 7816_C handles this for BAC and Unit 7816_P handles this for PACE.

If the eMRTD is BAC and PACE protected, PACE MUST be used.

Note: when accessing to EAC protected DG, Extended Access control MUST be granted.

3.4.1 Test Case 7816_D_1

Purpose	This test case verifies the SelectFile (EF.COM) command (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	BAC or PACE
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.COM SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 1E’. Send the following SelectFile APDU to the eMRTD. => ‘0C A4 02 0C <Lc>87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00’ 2. To verify that EF.COM is selected, send a valid ReadBinary APDU to the eMRTD. => ‘0C B0 00 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte 60 and the status bytes ‘90 00’.
Postconditions	EF.COM MUST be selected.

3.4.2 Test Case 7816_D_2

Purpose	This test case checks the robustness of the SelectFile command (invalid class byte).
Version	2.04
References	ICAO LDS [R1], [R2]
Profile	BAC or PACE
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.COM SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 1E’. Send the following APDU to the eMRTD. The class tag is set to the invalid value of ‘8F’. => ‘8F A4 02 0C <Lc>87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00’ 2. To verify that EF.COM is not selected, send a valid ReadBinary APDU to the eMRTD. => ‘0C B0 00 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error . 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Preconditions remain unchanged.

3.4.3 Test Case 7816_D_3

Purpose	This test case checks the robustness of the SelectFile command (invalid parameter P1).
Version	2.04
References	ICAO LDS [R1], [R2]
Profile	BAC or PACE
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.COM SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 1E’. Send the following SelectFile APDU to the eMRTD. The parameter P1 is set to the invalid value of ‘12’. => ‘0C A4 12 0C <Lc>87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00’ 2. To verify that EF.COM is not selected, send a valid ReadBinary APDU to the eMRTD. => ‘0C B0 00 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error . 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Preconditions remain unchanged.

3.4.4 Test Case 7816_D_4

Purpose	This test case checks the robustness of the SelectFile command (invalid parameter P2).
Version	2.04
References	ICAO LDS [R1], [R2]
Profile	BAC or PACE
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.COM SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 1E’. Send the following SelectFile APDU to the eMRTD. The parameter P2 is set to the invalid value of ‘1C’. => ‘0C A4 02 1C <Lc>87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00’ 2. To verify that EF.COM is not selected, send a valid ReadBinary APDU to the eMRTD. => ‘0C B0 00 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error . 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Preconditions remain unchanged.

3.4.5 Test Case 7816_D_5

Purpose	This test case checks the robustness of the SelectFile command (invalid Lc).
Version	2.04
References	ICAO LDS [R1], [R2]
Profile	BAC or PACE
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.COM SHALL be selected. Therefore, the cryptogram MUST contain the malformed file identifier ‘01 1E 01’ (Lc = ‘03’). Send the following SelectFile APDU to the eMRTD. => ‘0C A4 02 0C <Lc>87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00’ 2. To verify that EF.COM is not selected, send a valid ReadBinary APDU to the eMRTD. => ‘0C B0 00 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error . 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Preconditions remain unchanged.

3.4.6 Test Case 7816_D_6

Purpose	This test case verifies the SelectFile (EF.SOD) command (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	BAC or PACE
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.SOD SHALL be selected. Therefore, the cryptogram MUST contain the file identifier 01 1D. Send the following SelectFile APDU to the eMRTD. => ‘0C A4 02 0C <Lc>87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00’ 2. To verify that EF.SOD is selected, send a valid ReadBinary APDU to the eMRTD. => ‘0C B0 00 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘77’ and the status bytes ‘90 00’.
Postconditions	EF.SOD is selected.

3.4.7 Test Case 7816_D_7

Purpose	This test case verifies the SelectFile (EF.DG1) command (positive test).
Version	or PACE
References	ICAO LDS [R1], [R2]
Profile	BAC or PACE
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG1 SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 01’. Send the following SelectFile APDU to the eMRTD. => ‘0C A4 02 0C <Lc> 87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00’ 2. To verify that EF.DG1is selected, send a valid ReadBinary APDU to the eMRTD. => ‘0C B0 00 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘61’ and the status bytes ‘90 00’.
Postconditions	EF.DG1 is selected.

3.4.8 Test Case 7816_D_8

Purpose	This test case verifies the SelectFile (EF.DG2) command (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	BAC or PACE
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG2 SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 02’. Send the following SelectFile APDU to the eMRTD. => ‘0C A4 02 0C <Lc>87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00’ 2. To verify that EF.DG2 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘0C B0 00 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘75’ and the status bytes ‘90 00’.
Postconditions	EF.DG2 is selected.

3.4.9 Test Case 7816_D_9

Purpose	This test case verifies the SelectFile (EF.DG3) command (positive test).
Version	2.07
References	ICAO LDS [R1], [R2]
Profile	((BAC or PACE), DG3)
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG3 SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 03’. Send the following SelectFile APDU to the eMRTD. => ‘0C A4 02 0C <Lc>87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00’ 2. To verify that EF.DG3 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘0C B0 00 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘63’ and the status bytes ‘90 00’.
Postconditions	EF.DG3 is selected.

3.4.10 Test Case 7816_D_10

Purpose	This test case verifies the SelectFile (EF.DG4) command (positive test).
Version	2.07
References	ICAO LDS [R1], [R2]
Profile	((BAC or PACE), DG4)
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG4 SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 04’. Send the following SelectFile APDU to the eMRTD. => ‘0C A4 02 0C <Lc>87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00’ 2. To verify that EF.DG4 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘0C B0 00 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘76’ and the status bytes ‘90 00’.
Postconditions	EF.DG4 is selected.

3.4.11 Test Case 7816_D_11

Purpose	This test case verifies the SelectFile (EF.DG5) command (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), DG5
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG5 SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 05’. Send the following SelectFile APDU to the eMRTD. => ‘0C A4 02 0C <Lc>87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00’ 2. To verify that EF.DG5 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘0C B0 00 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘65’ and the status bytes ‘90 00’.
Postconditions	EF.DG5 is selected.

3.4.12 Test Case 7816_D_12

Purpose	This test case verifies the SelectFile (EF.DG6) command (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), DG6
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG6 SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 06’. Send the following SelectFile APDU to the eMRTD. => ‘0C A4 02 0C <Lc> 87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00’ 2. To verify that EF.DG6 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘0C B0 00 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘66’ and the status bytes ‘90 00’.
Postconditions	EF.DG6 is selected.

3.4.13 Test Case 7816_D_13

Purpose	This test case verifies the SelectFile (EF.DG7) command (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), DG7
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG7 SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 07’. Send the following SelectFile APDU to the eMRTD. => ‘0C A4 02 0C <Lc>87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00’ 2. To verify that EF.DG7 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘0C B0 00 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘67’ and the status bytes ‘90 00’.
Postconditions	EF.DG7 is selected.

3.4.14 Test Case 7816_D_14

Purpose	This test case verifies the SelectFile (EF.DG8) command (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), DG8
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG8 SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 08’. Send the following SelectFile APDU to the eMRTD. => ‘0C A4 02 0C <Lc>87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00’ 2. To verify that EF.DG8 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘0C B0 00 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘68’ and the status bytes ‘90 00’.
Postconditions	EF.DG8 is selected.

3.4.15 Test Case 7816_D_15

Purpose	This test case verifies the SelectFile (EF.DG9) command (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), DG9
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG9 SHALL be selected. Therefore, the cryptogram MUST contain the file identifier '01 09'. Send the following SelectFile APDU to the eMRTD. => '0C A4 02 0C <Lc>87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00' 2. To verify that EF.DG9 is selected, send a valid ReadBinary APDU to the eMRTD. => '0C B0 00 00 0D 97 01 01 8E 08 <checksum> 00'
Expected results	1. The eMRTD MUST return the status bytes '90 00'. 2. The eMRTD MUST return byte '69' and the status bytes '90 00'.
Postconditions	EF.DG9 is selected.

3.4.16 Test Case 7816_D_16

Purpose	This test case verifies the SelectFile (EF.DG10) command (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), DG10
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG10 SHALL be selected. Therefore, the cryptogram MUST contain the file identifier '01 0A'. Send the following SelectFile APDU to the eMRTD. => '0C A4 02 0C <Lc>87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00' 2. To verify that EF.DG10 is selected, send a valid ReadBinary APDU to the eMRTD. => '0C B0 00 00 0D 97 01 01 8E 08 <checksum> 00'
Expected results	1. The eMRTD MUST return the status bytes '90 00'. 2. The eMRTD MUST return byte '6A' and the status bytes '90 00'.
Postconditions	EF.DG10 is selected.

3.4.17 Test Case 7816_D_17

Purpose	This test case verifies the SelectFile (EF.DG11) command (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), DG11
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG11 SHALL be selected. Therefore, the cryptogram MUST contain the file identifier '01 0B'. Send the following SelectFile APDU to the eMRTD. => '0C A4 02 0C <Lc>87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00' 2. To verify that EF.DG11 is selected, send a valid ReadBinary APDU to the eMRTD. => '0C B0 00 00 0D 97 01 01 8E 08 <checksum> 00'
Expected results	1. The eMRTD MUST return the status bytes '90 00'. 2. The eMRTD MUST return byte '6B' and the status bytes '90 00'.
Postconditions	EF.DG11 is selected.

3.4.18 Test Case 7816_D_18

Purpose	This test case verifies the SelectFile (EF.DG12) command (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), DG12
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG12 SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 0C’. Send the following SelectFile APDU to the eMRTD. => ‘0C A4 02 0C <Lc>87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00’ 2. To verify that EF.DG12 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘0C B0 00 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘6C’ and the status bytes ‘90 00’.
Postconditions	EF.DG12 is selected.

3.4.19 Test Case 7816_D_19

Purpose	This test case verifies the SelectFile (EF.DG13) command (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), DG13
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG13 SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 0D’. Send the following SelectFile APDU to the eMRTD. => ‘0C A4 02 0C <Lc>87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00’ 2. To verify that EF.DG13 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘0C B0 00 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘6D’ and the status bytes ‘90 00’.
Postconditions	EF.DG13 is selected.

3.4.20 Test Case 7816_D_20

Purpose	This test case verifies the SelectFile (EF.DG14) command (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	BAC , (EAC or PACE or AA-ECDSA)
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG14 SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 0E’. Send the following SelectFile APDU to the eMRTD. => ‘0C A4 02 0C <Lc>87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00’ 2. To verify that EF.DG14 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘0C B0 00 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘6E’ and the status bytes ‘90 00’.
Postconditions	EF.DG14 is selected.

3.4.21 Test Case 7816_D_21

Purpose	This test case verifies the SelectFile (EF.DG15) command (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), AA
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG15 SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 0F’. Send the following SelectFile APDU to the eMRTD. => ‘0C A4 02 0C <Lc>87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00’ 2. To verify that EF.DG15 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘0C B0 00 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘6F’ and the status bytes ‘90 00’.
Postconditions	File EF.DG15 is selected.

3.4.22 Test Case 7816_D_22

Purpose	This test case verifies the SelectFile (EF.DG16) command (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), DG16
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG16 SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 10’. Send the following SelectFile APDU to the eMRTD. => ‘0C A4 02 0C <Lc>87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00’ 2. To verify that EF.DG16 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘0C B0 00 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘70’ and the status bytes ‘90 00’.
Postconditions	EF.DG16 is selected.

3.4.23 Test Case 7816_D_23

Purpose	This test case verifies the SelectFile command when the file to be selected does not exist.
Version	2.04
References	ICAO LDS [R1], [R2]
Profile	BAC or PACE
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. A not existing file SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘02 02’. Send the following SelectFile APDU to the eMRTD. => ‘0C A4 02 0C <Lc>87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Preconditions remain unchanged.

3.5 Unit Test ISO_7816_E – Protected ReadBinary Command

This unit verifies the implementation of the protected ReadBinary command.

The eMRTD MUST be BAC or/and PACE protected. For all test cases of unit test ISO_7816_E, basic access MUST be granted as tested in 7816_C_2 for BAC and 7816_P_1 for PACE. All APDUs MUST be correctly encoded for Secure Messaging and the eMRTD response MUST be correctly decoded again. The expected results of the test cases are plain text data after decoding the protected response APDU.

The tests in this unit do not test the secure messaging implementation including postconditions (e.g. SM termination); therefore, status bytes MAY be returned in secure messaging or without it. Unit 7816_C handles this for BAC and Unit 7816_P handles this for PACE.

If the eMRTD is BAC and PACE protected, PACE MUST be used.

Note: For the ReadBinary command in Secure Messaging mode, there is no clear definition in the ISO specification if the Le byte in DO ‘97’ = ‘00’. Test cases E_1 to E_3 and E_5 to E_22 use Le = ‘01’ in order to avoid unspecified EOF situations

Note: when accessing to protected DG by EAC, extended Access control MUST be granted.

3.5.1 Test Case 7816_E_1

Purpose	This test case verifies the ReadBinary command (w/o SFI) (positive test).
Version	2.04
References	ICAO LDS [R1], [R2]
Profile	BAC or PACE
Preconditions	The LDS application MUST be selected.
Test scenario	1. EF.COM SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 1E’. Send the following SelectFile APDU to the eMRTD. => ‘0C A4 02 0C <Lc>87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00’ 2. Send the ReadBinary APDU to the eMRTD and read the first bytes of EF.COM => ‘0C B0 00 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return response data ‘60’ and the status bytes ‘90 00’.
Postconditions	Preconditions remain unchanged.

3.5.2 Test Case 7816_E_2

Purpose	Test the robustness of the ReadBinary command (w/o SFI) (invalid class byte).
Version	2.04
References	ICAO LDS [R1], [R2]
Profile	BAC or PACE
Preconditions	The LDS application MUST be selected. This test case implicitly tests the SelectFile command; so it is required that the eMRTD has previously passed the SelectFile Test 7816_D_1, otherwise this test will fail.
Test scenario	1. EF.COM SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 1E’. Send the following SelectFile APDU to the eMRTD. => ‘0C A4 02 0C <Lc>87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00’ 2. Send the ReadBinary APDU to the eMRTD and read the first byte of EF.COM, The class byte is set to the invalid value of ‘8F’. => ‘8F B0 00 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Preconditions remain unchanged.

3.5.3 Test Case 7816_E_3

Purpose	Test the robustness of the ReadBinary command (w/o SFI) (offset beyond EOF).
Version	2.04
References	ICAO LDS [R1], [R2]
Profile	BAC or PACE
Preconditions	The LDS application MUST be selected. This test case implicitly tests the SelectFile command; so it is required that the eMRTD has previously passed the SelectFile Test 7816_D_1, otherwise this test will fail.
Test scenario	1. EF.COM SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 1E’. Send the following SelectFile APDU to the eMRTD. => ‘0C A4 02 0C <Lc> 87 <L_{87}> 01 <cryptogram> 8E 08 <checksum> 00’ 2. Send the ReadBinary APDU to the eMRTD. The offset is beyond the end of the EF.COM file. Note: Since the actual file on the eMRTD could be larger than necessary, the eMRTD may return valid data in this case. If this happens, the test may have to be repeated with an appropriated offset. => ‘0C B0 7F FF 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Preconditions remain unchanged.

3.5.4 Test Case 7816_E_4

Purpose	Test the robustness of the ReadBinary command (w/o SFI) (Le beyond EOF).
Version	2.04
References	ICAO LDS [R1], [R2]
Profile	BAC or PACE
Preconditions	The LDS application MUST be selected. This test case implicitly tests the SelectFile command; so it is required that the eMRTD has previously passed the SelectFile Test 7816_D_1, otherwise this test will fail.
Test scenario	1. EF.COM SHALL be selected. Therefore, the cryptogram MUST contain the file identifier ‘01 1E’. Send the following SelectFile APDU to the eMRTD. => ‘0C A4 02 0C <Lc>87 <L$_{87}$> 01 <cryptogram> 8E 08 <checksum> 00’ 2. Send the ReadBinary APDU to the eMRTD. The Le Byte requests more data than available in the EF.COM file Note: Since the actual file on the eMRTD could be larger than necessary, the eMRTD may return valid data in this case. If this happens, the test may have to be repeated with an appropriated offset. => ‘0C B0 00 00 0D 97 01 E0 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return status bytes ‘90 00’ or an ISO warning or an ISO checking error or ISO execution error .[3]
Postconditions	Preconditions remain unchanged.

3.5.5 Test Case 7816_E_5

Purpose	This test case verifies the ReadBinary command (EF.COM SFI) (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	BAC or PACE
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first byte of the EF.COM. => ‘0C B0 9E 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.COM is selected.

[3] Currently there is a task for TF1 and TF5 to define correct responses for this test case and publish them in the next DOC9303 supplement.

3.5.6 Test Case 7816_E_6

Purpose	This test case verifies the ReadBinary command (EF.SOD SFI) (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	BAC or PACE
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first byte of the EF.SOD. => ‘0C B0 9D 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.SOD is selected.

3.5.7 Test Case 7816_E_7

Purpose	This test case verifies the ReadBinary command (EF.DG1 SFI) (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	BAC or PACE
Preconditions	The LDS application MUST be selected. EF.DG1 MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first byte of the EF.DG1. => ‘0C B0 81 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG1 is selected.

3.5.8 Test Case 7816_E_8

Purpose	This test case verifies the ReadBinary command (EF.DG2 SFI) (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	BAC or PACE
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first byte of the EF.DG2. => ‘0C B0 82 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG2 is selected.

3.5.9 Test Case 7816_E_9

Purpose	This test case verifies the ReadBinary command (EF.DG3 SFI) (positive test).
Version	2.07
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), DG3
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first byte of the EF.DG3. => ‘0C B0 83 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG3 is selected.

3.5.10 Test Case 7816_E_10

Purpose	This test case verifies the ReadBinary command (EF.DG4 SFI) (positive test).
Version	2.07
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), DG4
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first byte of the EF.DG3. => ‘0C B0 84 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG4 is selected.

3.5.11 Test Case 7816_E_11

Purpose	This test case verifies the ReadBinary command (EF.DG5 SFI) (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), DG5
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first byte of the EF.DG5. => ‘0C B0 85 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG5 is selected.

3.5.12 Test Case 7816_E_12

Purpose	This test case verifies the ReadBinary command (EF.DG6 SFI) (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), DG6
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first byte of the EF.DG6. => ‘0C B0 86 00 0D 97 01 E0 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG6 is selected.

3.5.13 Test Case 7816_E_13

Purpose	This test case verifies the ReadBinary command (EF.DG7 SFI) (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), DG7
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first byte of the EF.DG7. => ‘0C B0 87 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG7 is selected.

3.5.14 Test Case 7816_E_14

Purpose	This test case verifies the ReadBinary command (EF.DG8 SFI) (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), DG8
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first byte of the EF.DG8. => ‘0C B0 88 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG8 is selected.

3.5.15 Test Case 7816_E_15

Purpose	This test case verifies the ReadBinary command (EF.DG9 SFI) (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), DG9
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first byte of the EF.DG9. => ‘0C B0 89 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG9 is selected.

3.5.16 Test Case 7816_E_16

Purpose	This test case verifies the ReadBinary command (EF.DG10 SFI) (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), DG10
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first byte of the EF.DG10. => ‘0C B0 8A 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG10 is selected.

3.5.17 Test Case 7816_E_17

Purpose	This test case verifies the ReadBinary command (EF.DG11 SFI) (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), DG11
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first byte of the EF.DG11. => ‘0C B0 8B 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG11 is selected.

3.5.18 Test Case 7816_E_18

Purpose	This test case verifies the ReadBinary command (EF.DG12 SFI) (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), DG12
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first byte of the EF.DG12. => ‘0C B0 8C 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG12 is selected.

3.5.19 Test Case 7816_E_19

Purpose	This test case verifies the ReadBinary command (EF.DG13 SFI) (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), DG13
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first byte of the EF.DG13. => ‘0C B0 8D 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG13 is selected.

3.5.20 Test Case 7816_E_20

Purpose	This test case verifies the ReadBinary command (EF.DG14 SFI) (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	BAC, (EAC or PACE or AA-ECDSA)
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first byte of the EF.DG14. => ‘0C B0 8E 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG14 is selected.

3.5.21 Test Case 7816_E_21

Purpose	This test case verifies the ReadBinary command (EF.DG15 SFI) (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), AA
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first byte of the EF.DG15. => ‘0C B0 8F 00 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG15 is selected.

3.5.22 Test Case 7816_E_22

Purpose	This test case verifies the ReadBinary command (EF.DG16 SFI) (positive test).
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	(BAC or PACE), DG16
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first byte of the EF.DG16. => ‘0C B0 ‘90 00’ 0D 97 01 01 8E 08 <checksum> 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG16 is selected.

3.6 Unit Test ISO_7816_F – Unprotected SelectFile Command

This unit verifies the implementation of the unprotected SelectFile command. It is only applicable to the plain profile.

3.6.1 Test Case 7816_F_1

Purpose	This test case verifies the SelectFile (EF.COM) command (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.COM SHALL be selected. Send the following SelectFile APDU to the eMRTD. => ‘00 A4 02 0C 02 01 1E’ 2. To verify that EF.COM is selected, send a valid ReadBinary APDU to the eMRTD. => ‘00 B0 00 00 01’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘60’ and the status bytes ‘90 00’.
Postconditions	File EF.COM is selected.

3.6.2 Test Case 7816_F_2

Purpose	This test case checks the robustness of the SelectFile command (invalid class byte).
Version	2.04
References	ICAO LDS [R1], [R2]
Profile	Plain
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.COM SHALL be selected. Send the following SelectFile APDU to the eMRTD. The class tag is set to the invalid value of ‘8F’. => ‘8F A4 02 0C 02 01 1E’ 2. To verify that EF.COM is not selected, send a valid ReadBinary APDU to the eMRTD. => ‘00 B0 00 00 01’
Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error . 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Preconditions remain unchanged.

3.6.3 Test Case 7816_F_3

Purpose	This test case checks the robustness of the SelectFile command (invalid parameter P1).
Version	2.04
References	ICAO LDS [R1], [R2]
Profile	Plain
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.COM SHALL be selected. Send the following SelectFile APDU to the eMRTD. The parameter P1 is set to the invalid value of ‘12’. => ‘00 A4 12 0C 02 01 1E’ 2. To verify that EF.COM is not selected, send a valid ReadBinary APDU to the eMRTD. => ‘00 B0 00 00 01’
Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error . 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Preconditions remain unchanged.

3.6.4 Test Case 7816_F_4

Purpose	This test case checks the robustness of the SelectFile command (invalid parameter P2).
Version	2.04
References	ICAO LDS [R1], [R2]
Profile	Plain
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.COM SHALL be selected. Send the following SelectFile APDU to the eMRTD. The parameter P2 is set to the invalid value of ‘1C’. => ‘00 A4 02 1C 02 01 1E’ 2. To verify that EF.COM is not selected, send a valid ReadBinary APDU to the eMRTD. => ‘00 B0 00 00 01’
Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error . 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Preconditions remain unchanged.

3.6.5 Test Case 7816_F_5

Purpose	This test case checks the robustness of the SelectFile command (Invalid Lc).
Version	2.04
References	ICAO LDS [R1], [R2]
Profile	Plain
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.COM SHALL be selected. Send the following SelectFile APDU to the eMRTD. The parameter Lc is set to ‘03’. => ‘00 A4 02 0C 03 01 1E’ 2. To verify that EF.COM is not selected, send a valid ReadBinary APDU to the eMRTD. => ‘00 B0 00 00 01’
Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error . 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Preconditions remain unchanged.

3.6.6 Test Case 7816_F_6

Purpose	This test case verifies the SelectFile (EF.SOD) command (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF. SOD SHALL be selected. Send the following APDU to the eMRTD. => ‘00 A4 02 0C 02 01 1D’ 2. To verify that EF.SOD is selected, send a valid ReadBinary APDU to the eMRTD. => ‘00 B0 00 00 01’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘77’ and the status bytes ‘90 00’.
Postconditions	File EF.SOD is selected.

3.6.7 Test Case 7816_F_7

Purpose	This test case verifies the SelectFile (EF.DG1) command (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG1 SHALL be selected. Send the following APDU to the eMRTD. => ‘00 A4 02 0C 02 01 01’ 2. To verify that EF.DG1 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘00 B0 00 00 01’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘61’ and the status bytes ‘90 00’.
Postconditions	File EF.DG1 is selected.

3.6.8 Test Case 7816_F_8

Purpose	This test case verifies the SelectFile (EF.DG2) command (positive test).

Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG2 SHALL be selected. Send the following APDU to the eMRTD. => ‘00 A4 02 0C 02 01 02’ 2. To verify that EF.DG2 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘00 B0 00 00 01’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘75’ and the status bytes ‘90 00’.
Postconditions	File EF.DG2 is selected.

3.6.9 Test Case 7816_F_9

Purpose	This test case verifies the SelectFile (EF.DG3) command (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, DG3
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG3 SHALL be selected. Send the following APDU to the eMRTD. => ‘00 A4 02 0C 02 01 03’ 2. To verify that EF.DG3 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘00 B0 00 00 01’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘63’ and the status bytes ‘90 00’.
Postconditions	File EF.DG3 is selected.

3.6.10 Test Case 7816_F_10

Purpose	This test case verifies the SelectFile (EF.DG4) command (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, DG4
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG4 SHALL be selected. Send the following APDU to the eMRTD. => ‘00 A4 02 0C 02 01 04’ 2. To verify that EF.DG4 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘00 B0 00 00 01’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘76’ and the status bytes ‘90 00’.
Postconditions	File EF.DG4 is selected.

3.6.11 Test Case 7816_F_11

Purpose	This test case verifies the SelectFile (EF.DG5) command (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, DG5
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG5 SHALL be selected. Send the following APDU to the eMRTD. => ‘00 A4 02 0C 02 01 05’ 2. To verify that EF.DG5 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘00 B0 00 00 01’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘65’ and the status bytes ‘90 00’.
Postconditions	File EF.DG5 is selected.

3.6.12 Test Case 7816_F_12

Purpose	This test case verifies the SelectFile (EF.DG6) command (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, DG6
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG6 SHALL be selected. Send the following APDU to the eMRTD. => ‘00 A4 02 0C 02 01 06’ 2. To verify that EF.DG6 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘00 B0 00 00 01’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘66’ and the status bytes ‘90 00’.
Postconditions	File EF.DG6 is selected.

3.6.13 Test Case 7816_F_13

Purpose	This test case verifies the SelectFile (EF.DG7) command (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, DG7
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG7 SHALL be selected. Send the following APDU to the eMRTD. => ‘00 A4 02 0C 02 01 07’ 2. To verify that EF.DG7 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘00 B0 00 00 01’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘67’ and the status bytes ‘90 00’.
Postconditions	File EF.DG7 is selected.

3.6.14 Test Case 7816_F_14

Purpose	This test case verifies the SelectFile (EF.DG8) command (positive test).
Version	1.1

References	ICAO LDS [R1], [R2]
Profile	Plain, DG8
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG8 SHALL be selected. Send the following APDU to the eMRTD. => ‘00 A4 02 0C 02 01 08’ 2. To verify that EF.DG8 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘00 B0 00 00 01’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘68’ and the status bytes ‘90 00’.
Postconditions	File EF.DG8 is selected.

3.6.15 Test Case 7816_F_15

Purpose	This test case verifies the SelectFile (EF.DG9) command (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, DG9
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG9 SHALL be selected. Send the following APDU to the eMRTD. => ‘00 A4 02 0C 02 01 09’ 2. To verify that EF.DG9 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘00 B0 00 00 01’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘69’ and the status bytes ‘90 00’.
Postconditions	File EF.DG9 is selected.

3.6.16 Test Case 7816_F_16

Purpose	This test case verifies the SelectFile (EF.DG10) command (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, DG10
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG10 SHALL be selected. Send the following APDU to the eMRTD. => ‘00 A4 02 0C 02 01 0A’ 2. To verify that EF.DG10 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘00 B0 00 00 01’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘6A’ and the status bytes ‘90 00’.
Postconditions	File EF.DG10 is selected.

3.6.17 Test Case 7816_F_17

Purpose	This test case verifies the SelectFile (EF.DG11) command (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, DG11
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG11 SHALL be selected. Send the following APDU to the eMRTD. => ‘00 A4 02 0C 02 01 0B’ 2. To verify that EF.DG11 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘00 B0 00 00 01’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘6B’ and the status bytes ‘90 00’.
Postconditions	File EF.DG11 is selected.

3.6.18 Test Case 7816_F_18

Purpose	This test case verifies the SelectFile (EF.DG12) command (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, DG12
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG12 SHALL be selected. Send the following APDU to the eMRTD. => ‘00 A4 02 0C 02 01 0C’ 2. To verify that EF.DG12 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘00 B0 00 00 01’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘6C’ and the status bytes ‘90 00’.
Postconditions	File EF.DG12 is selected.

3.6.19 Test Case 7816_F_19

Purpose	This test case verifies the SelectFile (EF.DG13) command (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, DG13
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG13 SHALL be selected. Send the following APDU to the eMRTD. => ‘00 A4 02 0C 02 01 0D’ 2. To verify that EF.DG13 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘00 B0 00 00 01’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘6D’ and the status bytes ‘90 00’.
Postconditions	File EF.DG13 is selected.

3.6.20 Test Case 7816_F_20

Purpose	This test case verifies the SelectFile (EF.DG14) command (positive test).
Version	2.07

References	ICAO LDS [R1], [R2]
Profile	Plain, AA-ECDSA
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG14 SHALL be selected. Send the following APDU to the eMRTD. => ‘00 A4 02 0C 02 01 0E’ 2. To verify that EF.DG14 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘00 B0 00 00 01’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘6E’ and the status bytes ‘90 00’.
Postconditions	File EF.DG14 is selected.

3.6.21 Test Case 7816_F_21

Purpose	This test case verifies the SelectFile (EF.DG15) command (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, AA
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG15 SHALL be selected. Send the following APDU to the eMRTD. => ‘00 A4 02 0C 02 01 0F’ 2. To verify that EF.DG15 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘00 B0 00 00 01’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘6F’ and the status bytes ‘90 00’.
Postconditions	File EF.DG15 is selected.

3.6.22 Test Case 7816_F_22

Purpose	This test case verifies the SelectFile (EF.DG16) command (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, DG16
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. EF.DG16 SHALL be selected. Send the following APDU to the eMRTD. => ‘00 A4 02 0C 02 01 10’ 2. To verify that EF.DG16 is selected, send a valid ReadBinary APDU to the eMRTD. => ‘00 B0 00 00 01’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return byte ‘70’ and the status bytes ‘90 00’.
Postconditions	File EF.DG16 is selected.

3.6.23 Test Case 7816_F_23

Purpose	This test case verifies the SelectFile command when the file to be selected does not exist.
Version	2.04
References	ICAO LDS [R1], [R2]
Profile	Plain
Preconditions	The LDS application is selected. An EF MUST NOT be selected.
Test scenario	1. A not existing file SHALL be selected. Send the following SelectFile APDU to the eMRTD. => ‘00 A4 02 0C 02 02 02’ 2. To verify that no file is selected, send a valid ReadBinary APDU to the eMRTD. => ‘00 B0 00 00 01’
Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error . 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Preconditions remain unchanged.

3.7 Unit Test ISO_7816_G – Unprotected ReadBinary Command

This unit verifies the implementation of the unprotected ReadBinary command. It is only applicable to the plain profile.

3.7.1 Test Case 7816_G_1

Purpose	This test case verifies the ReadBinary command (w/o SFI) (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain
Preconditions	The LDS application MUST be selected.
Test scenario	1. Send the following SelectFile APDU to the eMRTD. => '00 A4 02 0C 02 01 1E' 2. Send the ReadBinary APDU to the eMRTD, this will read the first bytes (256 at maximum) of the EF.COM => '00 B0 00 00 00'
Expected results	1. The eMRTD MUST return the status bytes '90 00'. 2. The eMRTD MUST return the status bytes '90 00' or an ISO warning.
Postconditions	Preconditions remain unchanged.

3.7.2 Test Case 7816_G_2

Purpose	Test the robustness of the ReadBinary command (w/o SFI) (invalid class byte).
Version	2.04
References	ICAO LDS [R1], [R2]
Profile	Plain
Preconditions	The LDS application is selected. This test case implicitly tests the SelectFile command; so it is required that the eMRTD has previously passed the SelectFile Test 7816_F_1, otherwise this test will fail.
Test scenario	1. Send the following SelectFile APDU to the eMRTD. => '00 A4 02 0C 02 01 1E' 2. Send the ReadBinary APDU to the eMRTD. The class byte is set to the invalid value of '8F'. => '8F B0 00 00 00'
Expected results	1. The eMRTD MUST return the status bytes '90 00'. 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Preconditions remain unchanged.

3.7.3 Test Case 7816_G_3

Purpose	Test the robustness of the ReadBinary command (w/o SFI) (offset beyond EOF).
Version	2.04
References	ICAO LDS [R1], [R2]
Profile	Plain
Preconditions	The LDS application MUST be selected. This test case implicitly tests the SelectFile command; so it is required that the eMRTD has previously passed the SelectFile Test 7816_F_1, otherwise this test will fail.
Test scenario	1. Send the following SelectFile APDU to the eMRTD. => ‘00 A4 02 0C 02 01 1E’ 2. Send the ReadBinary APDU to the eMRTD. The offset is beyond the end of the EF.COM file. Note: Since the actual file on the eMRTD could be larger than necessary, the eMRTD may return valid data in this case. If this happens, the test may have to be repeated with an appropriated offset. => ‘00 B0 7F FF 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return an ISO checking error or ISO execution error .
Postconditions	Preconditions remain unchanged.

3.7.4 Test Case 7816_G_4

Purpose	Test the robustness of the ReadBinary command (w/o SFI) (Le beyond EOF).
Version	2.04
References	ICAO LDS [R1], [R2]
Profile	Plain
Preconditions	The LDS application MUST be selected. This test case implicitly tests the SelectFile command; so it is required that the eMRTD has previously passed the SelectFile Test 7816_F_1, otherwise this test will fail.
Test scenario	1. Send the following SelectFile APDU to the eMRTD. => ‘00 A4 02 0C 02 01 1E’ 2. Send the ReadBinary APDU to the eMRTD. The Le Byte requests more data than available in the EF.COM file Note: Since the actual file on the eMRTD could be larger than necessary, the eMRTD may return valid data in this case. If this happens, the test may have to be repeated with an appropriated offset. => ‘00 B0 00 00 E0’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’. 2. The eMRTD MUST return status bytes ’90 00’, an ISO warning or an ISO checking error or ISO execution error .[4]
Postconditions	Preconditions remain unchanged.

[4] Currently there is a task for TF1 and TF5 to define correct responses for this test case and publish them in the next DOC9303 supplement.

3.7.5 Test Case 7816_G_5

Purpose	This test case verifies the ReadBinary command (EF.COM SFI) (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first bytes (256 bytes at maximum) of the EF.COM. => ‘00 B0 9E 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.COM is selected.

3.7.6 Test Case 7816_G_6

Purpose	This test case verifies the ReadBinary command (EF.SOD SFI) (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first bytes (256 bytes at maximum) of the EF.SOD. => ‘00 B0 9D 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.SOD is selected.

3.7.7 Test Case 7816_G_7

Purpose	This test case verifies the ReadBinary command (EF.DG1 SFI) (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first bytes (256 bytes at maximum) of the EF.DG1. => ‘00 B0 81 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG1 is selected.

3.7.8 Test Case 7816_G_8

Purpose	This test case verifies the ReadBinary command (EF.DG2 SFI) (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first bytes (256 bytes at maximum) of the EF.DG2. => ‘00 B0 82 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG2 is selected.

3.7.9 Test Case 7816_G_9

Purpose	This test case verifies the ReadBinary command (EF.DG3 SFI) (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, DG3
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first bytes (256 bytes at maximum) of the EF.DG3. => ‘00 B0 83 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG3 is selected.

3.7.10 Test Case 7816_G_10

Purpose	This test case verifies the ReadBinary command (EF.DG4 SFI) (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, DG4
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first bytes (256 bytes at maximum) of the EF.DG3. => ‘00 B0 84 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG4 is selected.

3.7.11 Test Case 7816_G_11

Purpose	This test case verifies the ReadBinary command (EF.DG5 SFI) (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, DG5
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first bytes (256 bytes at maximum) of the EF.DG5. => ‘00 B0 85 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG5 is selected.

3.7.12 Test Case 7816_G_12

Purpose	This test case verifies the ReadBinary command (EF.DG6 SFI) (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, DG6
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first bytes (256 bytes at maximum) of the EF.DG6. => ‘00 B0 86 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG6 is selected.

3.7.13 Test Case 7816_G_13

Purpose	This test case verifies the ReadBinary command (EF.DG7 SFI) (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, DG7
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first bytes (256 bytes at maximum) of the EF.DG7. => ‘00 B0 87 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG7 is selected.

3.7.14 Test Case 7816_G_14

Purpose	This test case verifies the ReadBinary command (EF.DG8 SFI) (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, DG8
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first bytes (256 bytes at maximum) of the EF.DG8. => ‘00 B0 88 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG8 is selected.

3.7.15 Test Case 7816_G_15

Purpose	This test case verifies the ReadBinary command (EF.DG9 SFI) (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, DG9
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first bytes (256 bytes at maximum) of the EF.DG9. => ‘00 B0 89 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG9 is selected.

3.7.16 Test Case 7816_G_16

Purpose	This test case verifies the ReadBinary command (EF.DG10 SFI) (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, DG10
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first bytes (256 bytes at maximum) of the EF.DG10. => ‘00 B0 8A 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG10 is selected.

3.7.17 Test Case 7816_G_17

Purpose	This test case verifies the ReadBinary command (EF.DG11 SFI) (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, DG11
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first bytes (256 bytes at maximum) of the EF.DG11. => ‘00 B0 8B 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG11 is selected.

3.7.18 Test Case 7816_G_18

Purpose	This test case verifies the ReadBinary command (EF.DG12 SFI) (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, DG12
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first bytes (256 bytes at maximum) of the EF.DG12. => ‘00 B0 8C 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG12 is selected.

3.7.19 Test Case 7816_G_19

Purpose	This test case verifies the ReadBinary command (EF.DG13 SFI) (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, DG13
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first bytes (256 bytes at maximum) of the EF.DG13. => ‘00 B0 8D 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG13 is selected.

3.7.20 Test Case 7816_G_20

Purpose	This test case verifies the ReadBinary command (EF.DG14 SFI) (positive test).
Version	2.07
References	ICAO LDS [R1], [R2]
Profile	Plain, AA-ECDSA
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first bytes (256 bytes at maximum) of the EF.DG14. => ‘00 B0 8E 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG14 is selected.

3.7.21 Test Case 7816_G_21

Purpose	This test case verifies the ReadBinary command (EF.DG15 SFI) (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, AA
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first bytes (256 bytes at maximum) of the EF.DG15. => ‘00 B0 8F 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG15 is selected.

3.7.22 Test Case 7816_G_22

Purpose	This test case verifies the ReadBinary command (EF.DG16 SFI) (positive test).
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	Plain, DG16
Preconditions	The LDS application MUST be selected. An EF MUST NOT be selected.
Test scenario	1. Send the ReadBinary APDU to the eMRTD, this will read the first bytes (256 bytes at maximum) of the EF.DG16. => ‘00 B0 90 00 00’
Expected results	1. The eMRTD MUST return the status bytes ‘90 00’.
Postconditions	EF.DG16 is selected.

3.8 Unit ISO7816_O[5] - Security Conditions for PACE-protected eMRTDs

This unit tests the security conditions of a PACE-protected eMRTD. It MUST NOT be possible to select and read the content of any present file. The tests of this unit try to access the files with an explicit SelectFile command, a ReadBinary command with implicit file selection via the short file identifier (SFI), and unsecured ReadBinary while access is granted.
The tests in this unit only apply to PACE-protected eMRTDs (profile PACE).
The tests in this unit do not test the SM implementation including postconditions (e.g. SM termination); therefore, status bytes MAY be returned in SM or without it. Unit 7816_P handles this.
In the following test cases, the term "PACE protocol is granted" means that the inspection system has successfully authenticated to the eMRTD. The first PACEInfo data structure in the EF.CardAccess has to be used.
Note that unsecured SelectApplication command in this Test Unit can return '6982' or '9000'. According to SAC, PACEv2 protocol is implemented in addition to Basic Access Control. The unsecured SelectApplication command returns '9000' in this case. eMRTD supporting only PACEv2 without Basic Access Control could return '6982' on unsecured SelectApplication command.
Note that if nothing is mentioned, unsecured context is applied.

Note: when accessing to protected DG by EAC, extended Access control MUST be granted.

3.8.1 Test case ISO7816_O_01

Purpose	Accessing the EF.COM file with explicit file selection
Version	2.04
References	[R7R7] TR-SAC §2.2
Profile	PACE
Preconditions	1. Reset the chip

Test scenario	1. Send the following SelectApplication APDU to the eMRTD
	`'00 A4 04 0C 07 A0 00 00 02 47 10 01'`
	2. Send the following Select APDU to the eMRTD
	`'00 A4 02 0C 02 01 1E'`

Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. 2. The eMRTD MUST return status bytes '69 82' If SelectApplication in step 1 returned '69 82', this step is skipped

[5] Test units H to N are defined in document TR-03105 Part3.2 v1.2

3.8.2 Test case ISO7816_O_02

Purpose	Accessing the EF.SOD file with explicit file selection
Version	2.04
References	[R7R7] TR-SAC §2.2
Profile	PACE
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `'00 A4 04 0C 07 A0 00 00 02 47 10 01'` 2. Send the following Select APDU to the eMRTD `'00 A4 02 0C 02 01 1D'`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. 2. The eMRTD MUST return status bytes '69 82' If SelectApplication in step 1 returned '69 82', this step is skipped

3.8.3 Test case ISO7816_O_03

Purpose	Accessing the EF.DG1 file with explicit file selection
Version	2.04
References	[R7R7] TR-SAC §2.2
Profile	PACE
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `'00 A4 04 0C 07 A0 00 00 02 47 10 01'` 2. Send the following Select APDU to the eMRTD `'00 A4 02 0C 02 01 01'`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. 2. The eMRTD MUST return status bytes '69 82' If SelectApplication in step 1 returned '69 82', this step is skipped

3.8.4 Test case ISO7816_O_04

Purpose	Accessing the EF.DG2 file with explicit file selection
Version	2.04
References	[R7R7] TR-SAC §2.2
Profile	PACE
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `‘00 A4 04 0C 07 A0 00 00 02 47 10 01’` 2. Send the following Select APDU to the eMRTD `‘00 A4 02 0C 02 01 02’`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.5 Test case ISO7816_O_05

Purpose	Accessing the EF.DG3 file with explicit file selection
Version	2.04
References	[R7R7] TR-SAC §2.2
Profile	(PACE, EAC, DG3) or (PACE, DG3)
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `‘00 A4 04 0C 07 A0 00 00 02 47 10 01’` 2. Send the following Select APDU to the eMRTD `‘00 A4 02 0C 02 01 03’`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.6 Test case ISO7816_O_06

Purpose	Accessing the EF.DG4 file with explicit file selection
Version	2.04
References	[R7R7] TR-SAC §2.2
Profile	(PACE, EAC, DG4) or (PACE, DG4)
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `‘00 A4 04 0C 07 A0 00 00 02 47 10 01’` 2. Send the following Select APDU to the eMRTD `‘00 A4 02 0C 02 01 04’`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.7 Test case ISO7816_O_07

Purpose	Accessing the EF.DG5 file with explicit file selection
Version	2.04
References	[R7R7] TR-SAC §2.2
Profile	PACE, DG5
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `‘00 A4 04 0C 07 A0 00 00 02 47 10 01’` 2. Send the following Select APDU to the eMRTD `‘00 A4 02 0C 02 01 05’`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.8 Test case ISO7816_O_08

Purpose	Accessing the EF.DG6 file with explicit file selection
Version	2.04
References	[R7R7] TR-SAC §2.2
Profile	PACE, DG6
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `‘00 A4 04 0C 07 A0 00 00 02 47 10 01’` 2. Send the following Select APDU to the eMRTD `‘00 A4 02 0C 02 01 06’`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.9 Test case ISO7816_O_09

Purpose	Accessing the EF.DG7 file with explicit file selection
Version	2.04
References	[R7R7] TR-SAC §2.2
Profile	PACE, DG7
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `‘00 A4 04 0C 07 A0 00 00 02 47 10 01’` 2. Send the following Select APDU to the eMRTD `‘00 A4 02 0C 02 01 07’`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.10 Test case ISO7816_O_10

Purpose	Accessing the EF.DG8 file with explicit file selection
Version	2.04
References	[R7R7] TR-SAC §2.2
Profile	PACE, DG8
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `‘00 A4 04 0C 07 A0 00 00 02 47 10 01’` 2. Send the following Select APDU to the eMRTD `‘00 A4 02 0C 02 01 08’`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.11 Test case ISO7816_O_11

Purpose	Accessing the EF.DG9 file with explicit file selection
Version	2.04
References	[R7R7] TR-SAC §2.2
Profile	PACE, DG9
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `‘00 A4 04 0C 07 A0 00 00 02 47 10 01’` 2. Send the following Select APDU to the eMRTD `‘00 A4 02 0C 02 01 09’`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.12 Test case ISO7816_O_12

Purpose	Accessing the EF.DG10 file with explicit file selection
Version	2.04
References	[R7R7] TR-SAC §2.2
Profile	PACE, DG10
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `'00 A4 04 0C 07 A0 00 00 02 47 10 01'` 2. Send the following Select APDU to the eMRTD `'00 A4 02 0C 02 01 0A'`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.13 Test case ISO7816_O_13

Purpose	Accessing the EF.DG11 file with explicit file selection
Version	2.04
References	[R7R7] TR-SAC §2.2
Profile	PACE, DG11
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `'00 A4 04 0C 07 A0 00 00 02 47 10 01'` 2. Send the following Select APDU to the eMRTD `'00 A4 02 0C 02 01 0B'`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.14 Test case ISO7816_O_14

Purpose	Accessing the EF.DG12 file with explicit file selection
Version	2.04
References	[R7R7] TR-SAC §2.2
Profile	PACE, DG12
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `'00 A4 04 0C 07 A0 00 00 02 47 10 01'` 2. Send the following Select APDU to the eMRTD `'00 A4 02 0C 02 01 0C'`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. 2. The eMRTD MUST return status bytes '69 82' If SelectApplication in step 1 returned '69 82', this step is skipped

3.8.15 Test case ISO7816_O_15

Purpose	Accessing the EF.DG13 file with explicit file selection
Version	2.04
References	[R7R7] TR-SAC §2.2
Profile	PACE, DG13
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `'00 A4 04 0C 07 A0 00 00 02 47 10 01'` 2. Send the following Select APDU to the eMRTD `'00 A4 02 0C 02 01 0D'`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. 2. The eMRTD MUST return status bytes '69 82' If SelectApplication in step 1 returned '69 82', this step is skipped

3.8.16 Test case ISO7816_O_16

Purpose	Accessing the EF.DG14 file with explicit file selection
Version	2.04
References	[R7R7] TR-SAC §2.2
Profile	PACE
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `‘00 A4 04 0C 07 A0 00 00 02 47 10 01’` 2. Send the following Select APDU to the eMRTD `‘00 A4 02 0C 02 01 0E’`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.17 Test case ISO7816_O_17

Purpose	Accessing the EF.DG15 file with explicit file selection
Version	2.04
References	[R7R7] TR-SAC §2.2
Profile	PACE, AA
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `‘00 A4 04 0C 07 A0 00 00 02 47 10 01’` 2. Send the following Select APDU to the eMRTD `‘00 A4 02 0C 02 01 0F’`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.18 Test case ISO7816_O_18

Purpose	Accessing the EF.DG16 file with explicit file selection
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE, DG16
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `‘00 A4 04 0C 07 A0 00 00 02 47 10 01’` 2. Send the following SelectApplication APDU to the eMRTD `‘00 A4 02 0C 02 01 10’`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.19 Test case ISO7816_O_19

Purpose	Accessing the EF.COM file with implicit file selection (ReadBinary with SFI)
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `‘00 A4 04 0C 07 A0 00 00 02 47 10 01’` 2. Send the following ReadBinary APDU to the eMRTD `‘00 B0 9E 00 00’`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. Since the read access is prohibited without prior access control authentication, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.20 Test case ISO7816_O_20

Purpose	Accessing the EF.SOD file with implicit file selection (ReadBinary with SFI)
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `'00 A4 04 0C 07 A0 00 00 02 47 10 01'` 2. Send the following ReadBinary APDU to the eMRTD `'00 B0 9D 00 00'`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. 2. Since the read access is prohibited without prior access control authentication, the response data field MUST be empty. The eMRTD MUST return status bytes '69 82' If SelectApplication in step 1 returned '69 82', this step is skipped

3.8.21 Test case ISO7816_O_21

Purpose	Accessing the EF.DG1 file with implicit file selection (ReadBinary with SFI)
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `'00 A4 04 0C 07 A0 00 00 02 47 10 01'` 2. Send the following ReadBinary APDU to the eMRTD `'00 B0 81 00 00'`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. 2. Since the read access is prohibited without prior access control authentication, the response data field MUST be empty. The eMRTD MUST return status bytes '69 82' If SelectApplication in step 1 returned '69 82', this step is skipped

3.8.22 Test case ISO7816_O_22

Purpose	Accessing the EF.DG2 file with implicit file selection (ReadBinary with SFI)
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `‘00 A4 04 0C 07 A0 00 00 02 47 10 01’` 2. Send the following ReadBinary APDU to the eMRTD `‘00 B0 82 00 00’`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. Since the read access is prohibited without prior access control authentication, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.23 Test case ISO7816_O_23

Purpose	Accessing the EF.DG3 file with implicit file selection (ReadBinary with SFI)
Version	2.04
References	[R7] TR-SAC §2.2
Profile	(PACE,EAC, DG3) or (PACE, DG3)
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `‘00 A4 04 0C 07 A0 00 00 02 47 10 01’` 2. Send the following ReadBinary APDU to the eMRTD `‘00 B0 83 00 00’`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. Since the read access is prohibited without prior access control authentication, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.24 Test case ISO7816_O_24

Purpose	Accessing the EF.DG4 file with implicit file selection (ReadBinary with SFI)
Version	2.04
References	[R7] TR-SAC §2.2
Profile	(PACE, EAC, DG4) or (PACE, DG4)
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `'00 A4 04 0C 07 A0 00 00 02 47 10 01'` 2. Send the following ReadBinary APDU to the eMRTD `'00 B0 84 00 00'`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. 2. Since the read access is prohibited without prior access control authentication, the response data field MUST be empty. The eMRTD MUST return status bytes '69 82' If SelectApplication in step 1 returned '69 82', this step is skipped

3.8.25 Test case ISO7816_O_25

Purpose	Accessing the EF.DG5 file with implicit file selection (ReadBinary with SFI)
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE, DG5
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `'00 A4 04 0C 07 A0 00 00 02 47 10 01'` 2. Send the following ReadBinary APDU to the eMRTD `'00 B0 85 00 00'`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. 2. Since the read access is prohibited without prior access control authentication, the response data field MUST be empty. The eMRTD MUST return status bytes '69 82' If SelectApplication in step 1 returned '69 82', this step is skipped

3.8.26 Test case ISO7816_O_26

Purpose	Accessing the EF.DG6 file with implicit file selection (ReadBinary with SFI)
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE, DG6
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `‘00 A4 04 0C 07 A0 00 00 02 47 10 01’` 2. Send the following ReadBinary APDU to the eMRTD `‘00 B0 86 00 00’`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. Since the read access is prohibited without prior access control authentication, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.27 Test case ISO7816_O_27

Purpose	Accessing the EF.DG7 file with implicit file selection (ReadBinary with SFI)
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE, DG7
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `‘00 A4 04 0C 07 A0 00 00 02 47 10 01’` 2. Send the following ReadBinary APDU to the eMRTD `‘00 B0 87 00 00’`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. Since the read access is prohibited without prior access control authentication, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.28 Test case ISO7816_O_28

Purpose	Accessing the EF.DG8 file with implicit file selection (ReadBinary with SFI)
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE, DG8
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `‘00 A4 04 0C 07 A0 00 00 02 47 10 01’` 2. Send the following ReadBinary APDU to the eMRTD `‘00 B0 88 00 00’`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. Since the read access is prohibited without prior access control authentication, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.29 Test case ISO7816_O_29

Purpose	Accessing the EF.DG9 file with implicit file selection (ReadBinary with SFI)
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE, DG9
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `‘00 A4 04 0C 07 A0 00 00 02 47 10 01’` 2. Send the following ReadBinary APDU to the eMRTD `‘00 B0 89 00 00’`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. Since the read access is prohibited without prior access control authentication, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.30 Test case ISO7816_O_30

Purpose	Accessing the EF.DG10 file with implicit file selection (ReadBinary with SFI)
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE, DG10
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `‘00 A4 04 0C 07 A0 00 00 02 47 10 01’` 2. Send the following ReadBinary APDU to the eMRTD `‘00 B0 8A 00 00’`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. Since the read access is prohibited without prior access control authentication, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.31 Test case ISO7816_O_31

Purpose	Accessing the EF.DG11 file with implicit file selection (ReadBinary with SFI)
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE, DG11
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `‘00 A4 04 0C 07 A0 00 00 02 47 10 01’` 2. Send the following ReadBinary APDU to the eMRTD `‘00 B0 8B 00 00’`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. Since the read access is prohibited without prior access control authentication, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.32 Test case ISO7816_O_32

Purpose	Accessing the EF.DG12 file with implicit file selection (ReadBinary with SFI)
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE, DG12
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `‘00 A4 04 0C 07 A0 00 00 02 47 10 01’` 2. Send the following ReadBinary APDU to the eMRTD `‘00 B0 8C 00 00’`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. Since the read access is prohibited without prior access control authentication, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.33 Test case ISO7816_O_33

Purpose	Accessing the EF.DG13 file with implicit file selection (ReadBinary with SFI)
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE, DG13
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `‘00 A4 04 0C 07 A0 00 00 02 47 10 01’` 2. Send the following ReadBinary APDU to the eMRTD `‘00 B0 8D 00 00’`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. 2. Since the read access is prohibited without prior access control authentication, the response data field MUST be empty. The eMRTD MUST return status bytes ‘69 82’ If SelectApplication in step 1 returned ‘69 82’, this step is skipped

3.8.34 Test case ISO7816_O_34

Purpose	Accessing the EF.DG14 file with implicit file selection (ReadBinary with SFI)
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `'00 A4 04 0C 07 A0 00 00 02 47 10 01'` 2. Send the following ReadBinary APDU to the eMRTD `'00 B0 8E 00 00'`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. 2. Since the read access is prohibited without prior access control authentication, the response data field MUST be empty. The eMRTD MUST return status bytes '69 82' If SelectApplication in step 1 returned '69 82', this step is skipped

3.8.35 Test case ISO7816_O_35

Purpose	Accessing the EF.DG15 file with implicit file selection (ReadBinary with SFI)
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE, AA
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `'00 A4 04 0C 07 A0 00 00 02 47 10 01'` 2. Send the following ReadBinary APDU to the eMRTD `'00 B0 8F 00 00'`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. 2. Since the read access is prohibited without prior access control authentication, the response data field MUST be empty. The eMRTD MUST return status bytes '69 82' If SelectApplication in step 1 returned '69 82', this step is skipped

3.8.36 Test case ISO7816_O_36

Purpose	Accessing the EF.DG16 file with implicit file selection (ReadBinary with SFI)
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE, DG16
Preconditions	1. Reset the chip
Test scenario	1. Send the following SelectApplication APDU to the eMRTD `'00 A4 04 0C 07 A0 00 00 02 47 10 01'` 2. Send the following ReadBinary APDU to the eMRTD `'00 B0 90 00 00'`
Expected results	1. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. 2. Since the read access is prohibited without prior access control authentication, the response data field MUST be empty. The eMRTD MUST return status bytes '69 82' If SelectApplication in step 1 returned '69 82', this step is skipped

3.8.37 Test case ISO7816_O_37

Purpose	Accessing the EF.COM file with ReadBinary. The test verifies the enforcement of SM after the PACE protocol has been performed successfully.
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE
Preconditions	1. The PACE protocol MUST have been performed using the MRZ-derived password. 2. The LDS application MUST have been selected
Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.COM encoded as a valid SM to the eMRTD `'0C B0 9E 00 0D 97 01 06 8E 08 <checksum> 00'` 2. Send the following Read Binary APDU as an unsecured APDU to the eMRTD `'00 B0 00 00 00'`
Expected results	1. The eMRTD MUST return status bytes '90 00' in a valid SM response. 2. The eMRTD MUST return an ISO checking error or ISO execution error

3.8.38 Test case ISO7816_O_38

Purpose	Accessing the EF.SOD file with ReadBinary. The test verifies the enforcement of SM after the PACE protocol has been performed successfully.
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE
Preconditions	1. The PACE protocol MUST have been performed using the MRZ-derived password. 2. The LDS application MUST have been selected

Test scenario	1. Send the following “Read Binary (SFI)” APDU for EF.SOD encoded as a valid SM to the eMRTD
	`‘0C B0 9D 00 0D 97 01 06 8E 08 <checksum> 00’`
	2. Send the following Read Binary APDU as an unsecured APDU to the eMRTD
	`‘00 B0 00 00 00’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’ in a valid SM response. 2. The eMRTD MUST return an ISO checking error or ISO execution error

3.8.39 Test case ISO7816_O_39

Purpose	Accessing the EF.DG1 file with ReadBinary. The test verifies the enforcement of SM after the PACE protocol has been performed successfully.
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE
Preconditions	1. The PACE protocol MUST have been performed using the MRZ-derived password. 2. The LDS application MUST have been selected

Test scenario	1. Send the following “Read Binary (SFI)” APDU for EF.DG1 encoded as a valid SM to the eMRTD
	`‘0C B0 81 00 0D 97 01 06 8E 08 <checksum> 00’`
	2. Send the following Read Binary APDU as an unsecured APDU to the eMRTD
	`‘00 B0 00 00 00’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’ in a valid SM response. 2. The eMRTD MUST return an ISO checking error or ISO execution error

3.8.40 Test case ISO7816_O_40

Purpose	Accessing the EF.DG2 file with ReadBinary. The test verifies the enforcement of SM after the PACE protocol has been performed successfully.
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE
Preconditions	1. The PACE protocol MUST have been performed using the MRZ-derived password. 2. The LDS application MUST have been selected

Test scenario	1. Send the following “Read Binary (SFI)” APDU for EF.DG2 encoded as a valid SM to the eMRTD
	`‘0C B0 82 00 0D 97 01 06 8E 08 <checksum> 00’`
	2. Send the following Read Binary APDU as an unsecured APDU to the eMRTD
	`‘00 B0 00 00 00’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’ in a valid SM response. 2. The eMRTD MUST return an ISO checking error or ISO execution error

3.8.41 Test case ISO7816_O_41

Purpose	Accessing the EF.DG3 file with ReadBinary. The test verifies the enforcement of SM after the PACE protocol has been performed successfully.
Version	2.04
References	[R7] TR-SAC §2.2
Profile	(PACE, EAC, DG3) or (PACE, DG3)
Preconditions	1. The PACE protocol MUST have been performed using the MRZ-derived password. 2. The LDS application MUST have been selected 3. The Extended Access Control MUST be granted if necessary

Test scenario	1. Send the following “Read Binary (SFI)” APDU for EF.DG3 encoded as a valid SM to the eMRTD
	`‘0C B0 83 00 0D 97 01 06 8E 08 <checksum> 00’`
	2. Send the following Read Binary APDU as an unsecured APDU to the eMRTD
	`‘00 B0 00 00 00’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’ in a valid SM response. 2. The eMRTD MUST return an ISO checking error or ISO execution error

3.8.42 Test case ISO7816_O_42

Purpose	Accessing the EF.DG4 file with ReadBinary. The test verifies the enforcement of SM after the PACE protocol has been performed successfully.
Version	2.04
References	[R7] TR-SAC §2.2
Profile	(PACE, EAC, DG4) or (PACE, DG4)
Preconditions	1. The PACE protocol MUST have been performed using the MRZ-derived password. 2. The LDS application MUST have been selected 3. The Extended Access Control MUST be granted if necessary

Test scenario	1. Send the following “Read Binary (SFI)” APDU for EF.DG4 encoded as a valid SM to the eMRTD
	`‘0C B0 84 00 0D 97 01 06 8E 08 <checksum> 00’`
	2. Send the following Read Binary APDU as an unsecured APDU to the eMRTD
	`‘00 B0 00 00 00’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’ in a valid SM response. 2. The eMRTD MUST return an ISO checking error or ISO execution error

3.8.43 Test case ISO7816_O_43

Purpose	Accessing the EF.DG5 file with ReadBinary. The test verifies the enforcement of SM after the PACE protocol has been performed successfully.
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE, DG5
Preconditions	1. The PACE protocol MUST have been performed using the MRZ-derived password. 2. The LDS application MUST have been selected

Test scenario	1. Send the following “Read Binary (SFI)” APDU for EF.DG5 encoded as a valid SM to the eMRTD
	`‘0C B0 85 00 0D 97 01 06 8E 08 <checksum> 00’`
	2. Send the following Read Binary APDU as an unsecured APDU to the eMRTD
	`‘00 B0 00 00 00’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’ in a valid SM response. 2. The eMRTD MUST return an ISO checking error or ISO execution error

3.8.44 Test case ISO7816_O_44

Purpose	Accessing the EF.DG6 file with ReadBinary. The test verifies the enforcement of SM after the PACE protocol has been performed successfully.
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE, DG6
Preconditions	1. The PACE protocol MUST have been performed using the MRZ-derived password. 2. The LDS application MUST have been selected

Test scenario	1. Send the following “Read Binary (SFI)” APDU for EF.DG6 encoded as a valid SM to the eMRTD
	`‘0C B0 86 00 0D 97 01 06 8E 08 <checksum> 00’`
	2. Send the following Read Binary APDU as an unsecured APDU to the eMRTD
	`‘00 B0 00 00 00’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’ in a valid SM response. 2. The eMRTD MUST return an ISO checking error or ISO execution error

3.8.45 Test case ISO7816_O_45

Purpose	Accessing the EF.DG7 file with ReadBinary. The test verifies the enforcement of SM after the PACE protocol has been performed successfully.
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE, DG7
Preconditions	1. The PACE protocol MUST have been performed using the MRZ-derived password. 2. The LDS application MUST have been selected

Test scenario	1. Send the following “Read Binary (SFI)” APDU for EF.DG7 encoded as a valid SM to the eMRTD
	`‘0C B0 87 00 0D 97 01 06 8E 08 <checksum> 00’`
	2. Send the following Read Binary APDU as an unsecured APDU to the eMRTD
	`‘00 B0 00 00 00’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’ in a valid SM response. 2. The eMRTD MUST return an ISO checking error or ISO execution error

3.8.46 Test case ISO7816_O_46

Purpose	Accessing the EF.DG8 file with ReadBinary. The test verifies the enforcement of SM after the PACE protocol has been performed successfully.
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE, DG8
Preconditions	1. The PACE protocol MUST have been performed using the MRZ-derived password. 2. The LDS application MUST have been selected

Test scenario	1. Send the following “Read Binary (SFI)” APDU for EF.DG8 encoded as a valid SM to the eMRTD
	`‘0C B0 88 00 0D 97 01 06 8E 08 <checksum> 00’`
	2. Send the following Read Binary APDU as an unsecured APDU to the eMRTD
	`‘00 B0 00 00 00’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’ in a valid SM response. 2. The eMRTD MUST return an ISO checking error or ISO execution error

3.8.47 Test case ISO7816_O_47

Purpose	Accessing the EF.DG9 file with ReadBinary. The test verifies the enforcement of SM after the PACE protocol has been performed successfully.
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE, DG9
Preconditions	1. The PACE protocol MUST have been performed using the MRZ-derived password. 2. The LDS application MUST have been selected

Test scenario	1. Send the following “Read Binary (SFI)” APDU for EF.DG9 encoded as a valid SM to the eMRTD
	`‘0C B0 89 00 0D 97 01 06 8E 08 <checksum> 00’`
	2. Send the following Read Binary APDU as an unsecured APDU to the eMRTD
	`‘00 B0 00 00 00’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’ in a valid SM response. 2. The eMRTD MUST return an ISO checking error or ISO execution error

3.8.48 Test case ISO7816_O_48

Purpose	Accessing the EF.DG10 file with ReadBinary. The test verifies the enforcement of SM after the PACE protocol has been performed successfully.
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE, DG10
Preconditions	1. The PACE protocol MUST have been performed using the MRZ-derived password. 2. The LDS application MUST have been selected

Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.DG10 encoded as a valid SM to the eMRTD
	`'0C B0 8A 00 0D 97 01 06 8E 08 <checksum> 00'`
	2. Send the following Read Binary APDU as an unsecured APDU to the eMRTD
	`'00 B0 00 00 00'`

Expected results	1. The eMRTD MUST return status bytes '90 00' in a valid SM response. 2. The eMRTD MUST return an ISO checking error or ISO execution error

3.8.49 Test case ISO7816_O_49

Purpose	Accessing the EF.DG11 file with ReadBinary. The test verifies the enforcement of SM after the PACE protocol has been performed successfully.
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE, DG11
Preconditions	1. The PACE protocol MUST have been performed using the MRZ-derived password. 2. The LDS application MUST have been selected

Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.DG11 encoded as a valid SM to the eMRTD
	`'0C B0 8B 00 0D 97 01 06 8E 08 <checksum> 00'`
	2. Send the following Read Binary APDU as an unsecured APDU to the eMRTD
	`'00 B0 00 00 00'`

Expected results	1. The eMRTD MUST return status bytes '90 00' in a valid SM response. 2. The eMRTD MUST return an ISO checking error or ISO execution error

3.8.50 Test case ISO7816_O_50

Purpose	Accessing the EF.DG12 file with ReadBinary. The test verifies the enforcement of SM after the PACE protocol has been performed successfully.
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE, DG12
Preconditions	1. The PACE protocol MUST have been performed using the MRZ-derived password. 2. The LDS application MUST have been selected

Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.DG12 encoded as a valid SM to the eMRTD
	`'0C B0 8C 00 0D 97 01 06 8E 08 <checksum> 00'`
	2. Send the following Read Binary APDU as an unsecured APDU to the eMRTD
	`'00 B0 00 00 00'`

Expected results	1. The eMRTD MUST return status bytes '90 00' in a valid SM response. 2. The eMRTD MUST return an ISO checking error or ISO execution error

3.8.51 Test case ISO7816_O_51

Purpose	Accessing the EF.DG13 file with ReadBinary. The test verifies the enforcement of SM after the PACE protocol has been performed successfully.
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE, DG13
Preconditions	1. The PACE protocol MUST have been performed using the MRZ-derived password. 2. The LDS application MUST have been selected

Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.DG13 encoded as a valid SM to the eMRTD
	`'0C B0 8D 00 0D 97 01 06 8E 08 <checksum> 00'`
	2. Send the following Read Binary APDU as an unsecured APDU to the eMRTD
	`'00 B0 00 00 00'`

Expected results	1. The eMRTD MUST return status bytes '90 00' in a valid SM response. 2. The eMRTD MUST return an ISO checking error or ISO execution error

3.8.52 Test case ISO7816_O_52

Purpose	Accessing the EF.DG14 file with ReadBinary. The test verifies the enforcement of SM after the PACE protocol has been performed successfully.
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE
Preconditions	1. The PACE protocol MUST have been performed using the MRZ-derived password. 2. The LDS application MUST have been selected

Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.DG14 encoded as a valid SM to the eMRTD
	`'0C B0 8E 00 0D 97 01 06 8E 08 <checksum> 00'`
	2. Send the following Read Binary APDU as an unsecured APDU to the eMRTD
	`'00 B0 00 00 00'`

Expected results	1. The eMRTD MUST return status bytes '90 00' in a valid SM response. 2. The eMRTD MUST return an ISO checking error or ISO execution error

3.8.53 Test case ISO7816_O_53

Purpose	Accessing the EF.DG15 file with ReadBinary. The test verifies the enforcement of SM after the PACE protocol has been performed successfully.
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE, AA
Preconditions	1. The PACE protocol MUST have been performed using the MRZ-derived password. 2. The LDS application MUST have been selected

Test scenario	1. Send the following "Read Binary (SFI)" APDU for EF.DG15 encoded as a valid SM to the eMRTD
	`'0C B0 8F 00 0D 97 01 06 8E 08 <checksum> 00'`
	2. Send the following Read Binary APDU as an unsecured APDU to the eMRTD
	`'00 B0 00 00 00'`

Expected results	1. The eMRTD MUST return status bytes '90 00' in a valid SM response. 2. The eMRTD MUST return an ISO checking error or ISO execution error

3.8.54 Test case ISO7816_O_54

Purpose	Accessing the EF.DG16 file with ReadBinary. The test verifies the enforcement of SM after the PACE protocol has been performed successfully.
Version	2.04
References	[R7] TR-SAC §2.2
Profile	PACE, DG16
Preconditions	1. The PACE protocol MUST have been performed using the MRZ-derived password. 2. The LDS application MUST have been selected

Test scenario	1. Send the following “Read Binary (SFI)” APDU for EF.DG16 encoded as a valid SM to the eMRTD
	`‘0C B0 90 00 0D 97 01 06 8E 08 <checksum> 00’`
	2. Send the following Read Binary APDU as an unsecured APDU to the eMRTD
	`‘00 B0 00 00 00’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’ in a valid SM response. 2. The eMRTD MUST return an ISO checking error or ISO execution error

3.9 Unit ISO7816_P – Password Authenticated Connection Establishment (PACEv2)

This unit checks the PACE implementation of the eMRTD. The complete PACE access mechanism is tested, including robustness tests with invalid input data.
Since the tests in this unit apply to PACE-protected eMRTDs, they are only mandatory for eMRTDs complying with the PACE profile.
In case of PACEv2 failure, [R7] doesn't define clearly the conditions of use of BAC mechanism. This context is out of the scope of this test unit.

PACE establishes SM between an eMRTD and an inspection system based on weak (short) passwords. It enables the eMRTD to verify that the inspection system is authorized to access stored data and has the following features:

- Strong session keys are provided independent of the strength of the password.
- The entropy of the password(s) used to authenticate the inspection system can be very low (e.g. 6 digits are sufficient in general).

PACE uses keys K_π derived from passwords. For globally interoperable machine readable travel documents the following two passwords and corresponding keys are available as follows:

- **MRZ:** The key $K_\pi = KDF_\pi(MRZ)$ is REQUIRED. It is derived from the Machine Readable Zone (MRZ) similar to Basic Access Control, i.e. the key is derived from the Document Number, the Date of Birth and the Date of Expiry.
- **CAN:** The key $K_\pi = KDF_\pi(CAN)$ is OPTIONAL. It is derived from the Card Access Number (CAN). The CAN is a number printed on the *front side* of the datapage

This unit MUST be executed for each PACE protocol indicated in the PACEInfo elements present in the EF.CardAccess of the eMRTD. Pre-conditions MUST be run with each PACEInfo elements.

Note that unsecured SelectApplication command in this Test Unit can return '6982' or '9000'. According to SAC, PACEv2 protocol is implemented in addition to Basic Access Control. The unsecured SelectApplication command returns '9000' in this case. eMRTD supporting only PACEv2 without Basic Access Control could return '6982' on unsecured SelectApplication command
Note that unsecured context is applied if nothing is mentioned.

3.9.1 Test case ISO7816_P_01

Purpose	Positive test with a valid PACE v2 protocol with MRZ password
Version	2.0
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the Chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`’00 86 00 00 <Lc> 7C <L7C> 85 <L85> <Authentication Token> <Le>’`
	6. To verify the chip's ability to start the SM with the session keys, the Select LDS command is sent to the chip under SM.
	`‘0C A4 04 0C <Lc> 87 <L87> 01 <Cryptogram> 8E 08 <Checksum> 00’`
	- <Cryptogram> contains eMRTD Application identifier ‘A0 00 00 02 47 10 01’ encrypted according to the SM being used

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return:
	`’7C <L7C> 84 <L84> <Ephemeral Public Key> 90 00’`
	5. The eMRTD MUST return:
	`’7C <L7C> 86 <L86> <Authentication Token> 90 00’`
	6. The eMRTD MUST return status bytes ’90 00’ in a valid SM response.

3.9.2 Test case ISO7816_P_02

Purpose	Positive test with a valid PACE v2 protocol with CAN password
Version	2.0
References	[R7] TR-SAC
Profile	PACE, PACE-CAN
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with CAN password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 02 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`’00 86 00 00 <Lc> 7C <L7C> 85 <L85> <Authentication Token> <Le>’`
	6. To verify the chip's ability to start the SM with the session keys, the Select LDS command is sent to the chip under SM.
	`‘0C A4 04 0C <Lc> 87 <L87> 01 <Cryptogram> 8E 08 <Checksum> 00’`
	- <Cryptogram> contains eMRTD Application identifier ‘A0 00 00 02 47 10 01’ encrypted according to the SM being used

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return:
	`’7C <L7C> 84 <L84> <Ephemeral Public Key> 90 00’`
	5. The eMRTD MUST return:
	`’7C <L7C> 86 <L86> <Authentication Token> 90 00’`
	6. The eMRTD MUST return status bytes ’90 00’ in a valid SM response.

3.9.3 Test case ISO7816_P_03

Purpose	Valid PACE v2 protocol with MRZ password, but afterwards command without

	SM is used
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`’00 86 00 00 <Lc> 7C <L7C> 85 <L85> <Authentication Token> <Le>’`
	6. Select LDS application under SM
	`‘0C A4 04 0C <Lc> 87 <L87> 01 <Cryptogram> 8E 08 <Checksum> 00’`
	- <Cryptogram> contains eMRTD Application identifier ‘A0 00 00 02 47 10 01’ encrypted according to the SM being used
	7. To verify the chip's ability to still require Secured APDU after performing valid PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return:
	`’7C <L7C> 84 <L84> <Ephemeral Public Key> 90 00’`
	5. The eMRTD MUST return:
	`’7C <L7C> 86 <L86> <Authentication Token> 90 00’`
	6. The eMRTD MUST return status bytes ’90 00’ in a valid SM response.
	7. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.4 Test case ISO7816_P_04

Purpose	MSE: Set AT command with invalid class byte
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’80 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Select LDS application
	4. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error 2. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning 3. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO checking error or ISO execution error . 4. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.5 Test case ISO7816_P_05

Purpose	MSE: Set AT command with an invalid data object tag
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 81 <L81> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Select LDS application
	4. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error 2. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning 3. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO checking error or ISO execution error . 4. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.6 Test case ISO7816_P_06

Purpose	MSE: Set AT command with an invalid PACE OID
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : {id-PACE 5}
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Select LDS application
	4. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error 2. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning 3. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO checking error or ISO execution error . 4. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.7 Test case ISO7816_P_07

Purpose	MSE: Set AT command with a PACE OID with tag ‘0x06’
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> {06 <LPACE OID> <PACE OID>} 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Select LDS application
	4. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error 2. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning 3. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO checking error or ISO execution error . 4. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.8 Test case ISO7816_P_08

Purpose	MSE: Set AT command with a bad reference password
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 <Invalid password identifier> 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Select LDS application
	4. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error or '90 00'. The actual security operation which must fail using the wrong password reference in the MSE:Set AT command may be performed in a subsequent command. 2. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning 3. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO checking error or ISO execution error . 4. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.9 Test case ISO7816_P_09

Purpose	MSE: Set AT command with a private key reference unknown from the chip
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- Use a private key reference unknown from the chip
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`’00 86 00 00 <Lc> 7C <L7C> 85 <L85> <Authentication Token> <Le>’`
	6. Select LDS application
	7. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error or '90 00' 2. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning or ‘90 00’. Steps 3 to 5 are skipped in case of ISO checking error or ISO execution error or ISO Warning. 3. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning or ‘90 00’. Steps 4 and 5 are skipped in case of ISO checking error or ISO execution error or ISO Warning. 4. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning or ‘90 00’. Step 5 is skipped in case of ISO checking error or ISO execution error or ISO Warning. 5. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning. 6. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO checking error or ISO execution error 7. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.10 Test case ISO7816_P_10

Removed in version v2.02

3.9.11 Test case ISO7816_P_11

Purpose	General Authenticate to get the encrypted nonce command with a bad dynamic authentication data tag
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 8C 00 <Le>’`
	3. Select LDS application
	4. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return ’90 00’ 2. The eMRTD MUST return an ISO checking error or ISO execution error 3. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO checking error or ISO execution error . 4. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.12 Test case ISO7816_P_12

Purpose	General Authenticate to get the encrypted nonce command without dynamic authentication data
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`'00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>'`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`'10 86 00 00 <Le>'`
	3. Select LDS application
	4. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`'<Unsecured command as defined in table 1>'`

Expected results	1. The eMRTD MUST return '90 00' 2. The eMRTD MUST return an ISO checking error or ISO execution error . 3. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. Next step is skipped in case of returning ISO checking error or ISO execution error . 4. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.13 Test case ISO7816_P_13

Purpose	General Authenticate to get the encrypted nonce command with an additional object data
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`'00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>'`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`'10 86 00 00 <Lc> 7C <L7C> 81 01 01 <Le>'`
	3. Select LDS application
	4. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`'<Unsecured command as defined in table 1>'`

Expected results	1. The eMRTD MUST return '90 00' 2. The eMRTD MUST return an ISO checking error or ISO execution error 3. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. Next step is skipped in case of returning ISO checking error or ISO execution error . 4. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.14 Test case ISO7816_P_14

Purpose	Check PACE v2 protocol with MRZ password after performing twice General Authenticate APDU to get the encrypted nonce.
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	4. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	Note:
	Mapping data MUST be computed according to second generated nonce.
	5. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	6. Send the given General Authenticate APDU to perform mutual authenticate:
	`’00 86 00 00 <Lc> 7C <L7C> 85 <L85> <Authentication Token> <Le>’`
	7. To verify the chip's ability to start the SM with the session keys, an arbitrary SM APDU is sent to the chip.
	`‘0C A4 04 0C <Lc> 87 <L87> 01 <Cryptogram> 8E 08 <Checksum> 00’`
	- <Cryptogram> contains eMRTD Application identifier ‘A0 00 00 02 47 10 01’ encrypted according to the SM being used

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’` with encrypted nonce different from step 2 or ISO checking error or ISO execution error . Remaining steps are skipped in case of error.[6]
	4. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	Note:

[6] Behavior depends on implementation

	In case of Integrated Mapping, <L_{82}> MUST be set to '00' and < Mapping Data> MUST be empty.
	5. The eMRTD MUST return:
	`'7C <L7C> 84 <L84> <Ephemeral Public Key> 90 00'`
	6. The eMRTD MUST return:
	`'7C <L7C> 86 <L86> <Authentication Token> 90 00'`
	7. The eMRTD MUST return status bytes '90 00' in a valid SM response.

3.9.15 Test case ISO7816_P_15

Purpose	General Authenticate APDU to map the nonce with a bad dynamic authentication data tag
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 8C <L8C> 81 <L81> <Mapping Data> <Le>’`
	4. Select LDS application
	5. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’ 2. The eMRTD MUST return: `’7C <L7C> 80 <L80> <encrypted nonce> 90 00’` 3. The eMRTD MUST return an ISO checking error or ISO execution error 4. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO checking error or ISO execution error . 5. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.16 Test case ISO7816_P_16

Purpose	General Authenticate APDU to map the nonce without a dynamic authentication data tag
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 81 <L81> <Mapping Data> <Le>’`
	4. Select LDS application
	5. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’ 2. The eMRTD MUST return: `’7C <L7C> 80 <L80> <encrypted nonce> 90 00’` 3. The eMRTD MUST return an ISO checking error or ISO execution error 4. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO checking error or ISO execution error . 5. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.17 Test case ISO7816_P_17

Purpose	General Authenticate APDU to map the nonce with a bad data object tag
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`'00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>'`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`'10 86 00 00 <Lc> 7C 00 <Le>'`
	3. Send the given General Authenticate APDU to map the nonce:
	`'10 86 00 00 <Lc> 7C <L7C> 82 <L82> <Mapping Data> <Le>'`
	4. Select LDS application
	5. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`'<Unsecured command as defined in table 1>'`

Expected results	1. The eMRTD MUST return status bytes '90 00' 2. The eMRTD MUST return: `'7C <L7C> 80 <L80> <encrypted nonce> 90 00'` 3. The eMRTD MUST return an ISO_Checking_Error or ISO execution error 4. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 5. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.18 Test case ISO7816_P_18

Purpose	General Authenticate APDU to map the nonce with an additional data object
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> {81 <L81> <Mapping Data> 87 01 01} <Le>’`
	4. Select LDS application
	5. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’ 2. The eMRTD MUST return: `’7C <L7C> 80 <L80> <encrypted nonce> 90 00’` 3. The eMRTD MUST return an ISO_Checking_Error or ISO execution error 4. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 5. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.19 Test case ISO7816_P_19

Purpose	General Authenticate APDU to perform key agreement with a bad dynamic authentication data tag
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`'00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>'`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`'10 86 00 00 <Lc> 7C 00 <Le>'`
	3. Send the given General Authenticate APDU to map the nonce:
	`'10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>'`
	4. Send the given General Authenticate APDU to perform key agreement:
	`'10 86 00 00 <Lc> 8C <L8C> 83 <L83> <Ephemeral Public Key> <Le>'`
	5. Select LDS application
	6. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`'<Unsecured command as defined in table 1>'`

Expected results	1. The eMRTD MUST return status bytes '90 00'
	2. The eMRTD MUST return:
	`'7C <L7C> 80 <L80> <encrypted nonce> 90 00'`
	3. The eMRTD MUST return:
	`'7C <L7C> 82 <L82> <Mapping Data> 90 00'`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to '00' and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return an ISO_Checking_Error or ISO execution error
	5. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error.
	6. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.20 Test case ISO7816_P_20

Purpose	General Authenticate APDU to perform key agreement without dynamic authentication data tag
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 83 <L83> <Ephemeral Public Key> <Le>’`
	5. Select LDS application
	6. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return an ISO_Checking_Error or ISO execution error
	5. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error.
	6. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.21 Test case ISO7816_P_21

Purpose	General Authenticate APDU to perform key agreement with a bad data object tag
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`'00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>'`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`'10 86 00 00 <Lc> 7C 00 <Le>'`
	3. Send the given General Authenticate APDU to map the nonce:
	`'10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>'`
	4. Send the given General Authenticate APDU to perform key agreement:
	`'10 86 00 00 <Lc> 7C <L7C> 84 <L84> <Ephemeral Public Key> <Le>'`
	5. Select LDS application
	6. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`'<Unsecured command as defined in table 1>'`

Expected results	1. The eMRTD MUST return status bytes '90 00'
	2. The eMRTD MUST return:
	`'7C <L7C> 80 <L80> <encrypted nonce> 90 00'`
	3. The eMRTD MUST return:
	`'7C <L7C> 82 <L82> <Mapping Data> 90 00'`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to '00' and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return an ISO_Checking_Error or ISO execution error
	5. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error.
	6. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.22 Test case ISO7816_P_22

Purpose	General Authenticate APDU to perform key agreement with an additional data object tag
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> {83 <L83> <Ephemeral Public Key> 84 01 01} <Le>’`
	5. Select LDS application
	6. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return an ISO_Checking_Error or ISO execution error
	5. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error.
	6. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.23 Test case ISO7816_P_23

Purpose	General Authenticate APDU to perform key agreement with invalid ephemeral public key (different key size)
Version	2.04
References	[R7] TR-SAC
Profile	PACE, PACE-ECDH
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	- The ephemeral public key MUST be generated with domain parameters specifying a different key size (e.g. for a 224 bit key after mapping the nonce, a 192 bit ephemeral key pair is created)
	5. Select LDS application
	6. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return an ISO_Checking_Error or ISO execution error
	5. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error.
	6. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.24 Test case ISO7816_P_24

Purpose	General Authenticate APDU to perform key agreement providing a (0,0) public

	key
Version	2.04
References	[R7] TR-SAC
Profile	PACE, PACE-ECDH
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`'00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>'`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`'10 86 00 00 <Lc> 7C 00 <Le>'`
	3. Send the given General Authenticate APDU to map the nonce:
	`'10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>'`
	4. Send the given General Authenticate APDU to perform key agreement:
	`'10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>'`
	- The public key has to be coded as ‘04||x||y’ where both x and y have a size according to the prime, but filled with ‘00’
	5. Select LDS application
	6. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`'<Unsecured command as defined in table 1>'`

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`'7C <L7C> 80 <L80> <encrypted nonce> 90 00'`
	3. The eMRTD MUST return:
	`'7C <L7C> 82 <L82> <Mapping Data> 90 00'`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return an ISO_Checking_Error or ISO execution error
	5. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error.
	6. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.25 Test case ISO7816_P_25

Purpose	General Authenticate APDU to perform key agreement - test borderline cases for x- and y- coordinates (small x coordinate)
Version	2.02
References	[R7] TR-SAC

Profile	PACE , PACE-ECDH
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	- Use an ephemeral public key with an x-coordinate requiring at least one byte less than the length of P. Pad with leading zero bytes.
	Generate key pairs at random until a public key satisfying the constraint is obtained
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`’00 86 00 00 <Lc> 7C <L7C> 85 <L85> <Authentication Token> <Le>’`
	6. To verify the chip's ability to start the SM with the session keys, the Select LDS command is sent to the chip under SM.
	`‘0C A4 04 0C <Lc> 87 <L87> 01 <Cryptogram> 8E 08 <Checksum> 00’`
	- <Cryptogram> contains eMRTD Application identifier ‘A0 00 00 02 47 10 01’ encrypted according to the SM being used

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return:
	`’7C <L7C> 84 <L84> <Ephemeral Public Key> 90 00’`
	5. The eMRTD MUST return:
	`’7C <L7C> 86 <L86> <Authentication Token> 90 00’`
	6. The eMRTD MUST return status bytes ’90 00’ in a valid SM response.

3.9.26 Test case ISO7816_P_26

Purpose	General Authenticate APDU to perform key agreement - test borderline cases for x- and y- coordinates (large x coordinate)
Version	2.0

References	[R7] TR-SAC
Profile	PACE, PACE-ECDH
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`'00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>'`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`'10 86 00 00 <Lc> 7C 00 <Le>'`
	3. Send the given General Authenticate APDU to map the nonce:
	`'10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>'`
	4. Send the given General Authenticate APDU to perform key agreement:
	`'10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>'`
	- Use a ephemeral public key with an x-coordinate having its highest bit set to 1
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`'00 86 00 00 <Lc> 7C <L7C> 85 <L85> <Authentication Token> <Le>'`
	6. To verify the chip's ability to start the SM with the session keys, the Select LDS command is sent to the chip under SM.
	`'0C A4 04 0C <Lc> 87 <L87> 01 <Cryptogram> 8E 08 <Checksum> 00'`
	- <Cryptogram> contains eMRTD Application identifier 'A0 00 00 02 47 10 01' encrypted according to the SM being used

Expected results	1. The eMRTD MUST return status bytes '90 00'
	2. The eMRTD MUST return:
	`'7C <L7C> 80 <L80> <encrypted nonce> 90 00'`
	3. The eMRTD MUST return:
	`'7C <L7C> 82 <L82> <Mapping Data> 90 00'`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to '00' and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return:
	`'7C <L7C> 84 <L84> <Ephemeral Public Key> 90 00'`
	5. The eMRTD MUST return:
	`'7C <L7C> 86 <L86> <Authentication Token> 90 00'`
	6. The eMRTD MUST return status bytes '90 00' in a valid SM response.

3.9.27 Test case ISO7816_P_27

Purpose	General Authenticate APDU to perform key agreement - test borderline cases for x- and y- coordinates (small y coordinate)
Version	2.02
References	[R7] TR-SAC

Profile	PACE , PACE-ECDH
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`'00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>'`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`'10 86 00 00 <Lc> 7C 00 <Le>'`
	3. Send the given General Authenticate APDU to map the nonce:
	`'10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>'`
	4. Send the given General Authenticate APDU to perform key agreement:
	`'10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>'`
	- Use an ephemeral public key with an y-coordinate requiring at least one byte less than the length of P. Pad with leading zero bytes.
	Generate key pairs at random until a public key satisfying the constraint is obtained
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`'00 86 00 00 <Lc> 7C <L7C> 85 <L85> <Authentication Token> <Le>'`
	6. To verify the chip's ability to start the SM with the session keys, the Select LDS command is sent to the chip under SM.
	`'0C A4 04 0C <Lc> 87 <L87> 01 <Cryptogram> 8E 08 <Checksum> 00'`
	- <Cryptogram> contains eMRTD Application identifier 'A0 00 00 02 47 10 01' encrypted according to the SM being used

Expected results	1. The eMRTD MUST return status bytes '90 00'
	2. The eMRTD MUST return:
	`'7C <L7C> 80 <L80> <encrypted nonce> 90 00'`
	3. The eMRTD MUST return:
	`'7C <L7C> 82 <L82> <Mapping Data> 90 00'`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to '00' and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return:
	`'7C <L7C> 84 <L84> <Ephemeral Public Key> 90 00'`
	5. The eMRTD MUST return:
	`'7C <L7C> 86 <L86> <Authentication Token> 90 00'`
	6. The eMRTD MUST return status bytes '90 00' in a valid SM response

3.9.28 Test case ISO7816_P_28

Purpose	General Authenticate APDU to perform key agreement - test borderline cases for x- and y- coordinates (large y coordinate)
Version	2.0
References	[R7] TR-SAC
Profile	PACE , PACE-ECDH
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	- Use a ephemeral public key with an y-coordinate having its highest bit set to 1
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`’00 86 00 00 <Lc> 7C <L7C> 85 <L85> <Authentication Token> <Le>’`
	6. To verify the chip's ability to start the SM with the session keys, the Select LDS command is sent to the chip under SM.
	`‘0C A4 04 0C <Lc> 87 <L87> 01 <Cryptogram> 8E 08 <Checksum> 00’`
	- <Cryptogram> contains eMRTD Application identifier ‘A0 00 00 02 47 10 01’ encrypted according to the SM being used

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return:
	`’7C <L7C> 84 <L84> <Ephemeral Public Key> 90 00’`
	5. The eMRTD MUST return:
	`’7C <L7C> 86 <L86> <Authentication Token> 90 00’`
	6. The eMRTD MUST return status bytes ’90 00’ in a valid SM response.

3.9.29 Test case ISO7816_P_29

Purpose	General Authenticate APDU to perform key agreement – value higher than the prime
Version	2.04
References	[R7] TR-SAC
Profile	PACE, PACE-DH
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`'00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>'`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`'10 86 00 00 <Lc> 7C 00 <Le>'`
	3. Send the given General Authenticate APDU to map the nonce:
	`'10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>'`
	4. Send the given General Authenticate APDU to perform key agreement:
	`'10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>'`
	- Use an ephemeral public key with a wrong value (value larger than the Prime)
	ephemeral public key = prime p + 1
	5. Select LDS application
	6. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`'<Unsecured command as defined in table 1>'`

Expected results	1. The eMRTD MUST return status bytes '90 00'
	2. The eMRTD MUST return:
	`'7C <L7C> 80 <L80> <encrypted nonce> 90 00'`
	3. The eMRTD MUST return:
	`'7C <L7C> 82 <L82> <Mapping Data> 90 00'`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to '00' and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return an ISO_Checking_Error or ISO execution error
	5. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error.
	6. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.30 Test case ISO7816_P_30

Purpose	General Authenticate APDU to perform key agreement – wrong point (value does not belong to the curve)
Version	2.04
References	[R7] TR-SAC
Profile	PACE, PACE-ECDH
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	- Use an ephemeral public key with a wrong point (value does not belong to the curve)
	5. Select LDS application
	6. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return an ISO_Checking_Error or ISO execution error
	5. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error.
	6. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.31 Test case ISO7816_P_31

Purpose	General Authenticate APDU to perform Mutual Authenticate with a bad dynamic authentication data tag
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`’00 86 00 00 <Lc> 8C <L7C> 85 <L85> <Authentication Token> <Le>’`
	6. To verify the chip's ability to reject Secured APDU after performing incomplete PACE v2 protocol, an arbitrary secured APDU is sent to the chip.
	`‘0C A4 04 0C <Lc> 87 <L87> 01 <Cryptogram> 8E 08 <Checksum> 00’`
	- <Cryptogram> contains eMRTD Application identifier ‘A0 00 00 02 47 10 01’ encrypted according to the SM being used

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return:
	`’7C <L7C> 84 <L84> <Ephemeral Public Key> 90 00’`
	5. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning
	6. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response

3.9.32 Test case ISO7816_P_32

Purpose	General Authenticate APDU to perform Mutual Authenticate without dynamic authentication data tag
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`’00 86 00 00 <Lc> 85 <L85> <Authentication Token> <Le>’`
	6. To verify the chip's ability to reject Secured APDU after performing incomplete PACE v2 protocol, an arbitrary secured APDU is sent to the chip.
	`‘0C A4 04 0C <Lc> 87 <L87> 01 <Cryptogram> 8E 08 <Checksum> 00’`
	- <Cryptogram> contains eMRTD Application identifier ‘A0 00 00 02 47 10 01’ encrypted according to the SM being used

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return:
	`’7C <L7C> 84 <L84> <Ephemeral Public Key> 90 00’`
	5. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning
	6. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response

3.9.33 Test case ISO7816_P_33

Purpose	General Authenticate APDU to perform Mutual Authenticate with a bad data object tag
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`’00 86 00 00 <Lc> 7C <L7C> 86 <L86> <Authentication Token> <Le>’`
	6. To verify the chip's ability to reject Secured APDU after performing incomplete PACE v2 protocol, an arbitrary secured APDU is sent to the chip.
	`‘0C A4 04 0C <Lc> 87 <L87> 01 <Cryptogram> 8E 08 <Checksum> 00’`
	- <Cryptogram> contains eMRTD Application identifier ‘A0 00 00 02 47 10 01’ encrypted according to the SM being used

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return:
	`’7C <L7C> 84 <L84> <Ephemeral Public Key> 90 00’`
	5. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning
	6. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response

3.9.34 Test case ISO7816_P_34

Purpose	General Authenticate APDU to perform Mutual Authenticate with an additional data object tag
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`’00 86 00 00 <Lc> 7C <L7C> {85 <L85> <Authentication Token> 86 01 01} <Le>’`
	6. To verify the chip's ability to reject Secured APDU after performing incomplete PACE v2 protocol, an arbitrary secured APDU is sent to the chip.
	`‘0C A4 04 0C <Lc> 87 <L87> 01 <Cryptogram> 8E 08 <Checksum> 00’`
	- <Cryptogram> contains eMRTD Application identifier ‘A0 00 00 02 47 10 01’ encrypted according to the SM being used

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return:
	`’7C <L7C> 84 <L84> <Ephemeral Public Key> 90 00’`
	5. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning
	6. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response

3.9.35 Test case ISO7816_P_35

Purpose	General Authenticate APDU to perform Mutual Authenticate with a wrong authentication token
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`’00 86 00 00 <Lc> 7C <L7C> 85 <L85> <Authentication Token> <Le>’`
	- Replace the last byte of the Authentication token by its complementary.
	ex: replace 0x00 by 0xFF
	6. To verify the chip's ability to reject Secured APDU after performing incomplete PACE v2 protocol, an arbitrary secured APDU is sent to the chip.
	`‘0C A4 04 0C <Lc> 87 <L87> 01 <Cryptogram> 8E 08 <Checksum> 00’`
	- <Cryptogram> contains eMRTD Application identifier ‘A0 00 00 02 47 10 01’ encrypted according to the SM being used

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return:
	`’7C <L7C> 84 <L84> <Ephemeral Public Key> 90 00’`
	5. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning
	6. The eMRTD MUST return an ISO checking error or ISO execution error

	in an unsecured response

3.9.36 Test case ISO7816_P_36

Purpose	General Authenticate APDU to perform Mutual Authenticate with a shorter authentication token
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`’00 86 00 00 <Lc> 7C <L7C> 85 <L85> <Authentication Token> <Le>’`
	- Remove the last byte of the Authentication token. `<L85> must be correctly computed`
	6. To verify the chip's ability to reject Secured APDU after performing incomplete PACE v2 protocol, an arbitrary secured APDU is sent to the chip.
	`‘0C A4 04 0C <Lc> 87 <L87> 01 <Cryptogram> 8E 08 <Checksum> 00’`
	- <Cryptogram> contains eMRTD Application identifier ‘A0 00 00 02 47 10 01’ encrypted according to the SM being used

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return:
	`’7C <L7C> 84 <L84> <Ephemeral Public Key> 90 00’`
	5. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning

	6. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response

3.9.37 Test case ISO7816_P_37

Purpose	General Authenticate to get the encrypted nonce command with invalid class byte
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`’90 86 00 00 <Lc> 7C 00 <Le>’`
	3. Select LDS application
	4. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return ’90 00’ 2. The eMRTD MUST return an ISO_Checking_Error or ISO execution error 3. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 4. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.38 Test case ISO7816_P_38

Purpose	General Authenticate APDU to map the nonce with invalid class byte
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`'00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>'`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`'10 86 00 00 <Lc> 7C 00 <Le>'`
	3. Send the given General Authenticate APDU to map the nonce:
	`'90 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>'`
	4. Select LDS application
	5. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`'<Unsecured command as defined in table 1>'`

Expected results	1. The eMRTD MUST return status bytes '90 00' 2. The eMRTD MUST return: `'7C <L7C> 80 <L80> <encrypted nonce> 90 00'` 3. The eMRTD MUST return an ISO_Checking_Error or ISO execution error 4. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 5. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.39 Test case ISO7816_P_39

Purpose	General Authenticate APDU to perform key agreement with invalid class byte
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Send the given General Authenticate APDU to perform key agreement:
	`’90 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	5. Select LDS application
	6. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return an ISO_Checking_Error or ISO execution error
	5. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error.
	6. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.40 Test case ISO7816_P_40

Purpose	General Authenticate APDU to perform Mutual Authenticate with invalid class byte
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`’80 86 00 00 <Lc> 7C <L7C> 85 <L85> <Authentication Token> <Le>’`
	6. To verify the chip's ability to reject Secured APDU after performing incomplete PACE v2 protocol, an arbitrary secured APDU is sent to the chip.
	`‘0C A4 04 0C <Lc> 87 <L87> 01 <Cryptogram> 8E 08 <Checksum> 00’`
	- <Cryptogram> contains eMRTD Application identifier ‘A0 00 00 02 47 10 01’ encrypted according to the SM being used

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return:
	`’7C <L7C> 84 <L84> <Ephemeral Public Key> 90 00’`
	5. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning
	6. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response

3.9.41 Test case ISO7816_P_41

Purpose	General Authenticate APDU to map the nonce with invalid ephemeral public key (different key size)
Version	2.04
References	[R7] TR-SAC
Profile	PACE, PACE-ECDH, PACE-GM
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`'00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>'`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`'10 86 00 00 <Lc> 7C 00 <Le>'`
	3. Send the given General Authenticate APDU to map the nonce:
	`'10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>'`
	- The ephemeral public key MUST be generated with domain parameters specifying a different key size (e.g. if a 224 bit key is required, a 192 bit ephemeral key pair is created)
	4. Select LDS application
	5. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`'<Unsecured command as defined in table 1>'`

Expected results	1. The eMRTD MUST return status bytes '90 00' 2. The eMRTD MUST return: `'7C <L7C> 80 <L80> <encrypted nonce> 90 00'` 3. The eMRTD MUST return an ISO_Checking_Error or ISO execution error 4. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 5. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.42 Test case ISO7816_P_42

Purpose	General Authenticate APDU to map the nonce providing a (0,0) public key
Version	2.04
References	[R7] TR-SAC
Profile	PACE, PACE-ECDH, PACE-GM
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	- The public key has to be coded as ‘04||x||y’ where both x and y have a size according to the prime, but filled with ‘00’
	4. Select LDS application
	5. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’ 2. The eMRTD MUST return: `’7C <L7C> 80 <L80> <encrypted nonce> 90 00’` 3. The eMRTD MUST return an ISO_Checking_Error or ISO execution error 4. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 5. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.43 Test case ISO7816_P_43

Purpose	General Authenticate APDU to map the nonce - test borderline cases for x- and y-coordinates (small x coordinate)
Version	2.0
References	[R7] TR-SAC
Profile	PACE , PACE-ECDH, PACE-GM
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	- Use an ephemeral public key with an x-coordinate requiring at least one byte less than the fewest bytes that can represent [$\log_{256}$ q]. Pad with zero bytes. (For details on q see [R8])
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`’00 86 00 00 <Lc> 7C <L7C> 85 <L85> <Authentication Token> <Le>’`
	6. To verify the chip's ability to start the SM with the session keys, the Select LDS command is sent to the chip under SM.
	`‘0C A4 04 0C <Lc> 87 <L87> 01 <Cryptogram> 8E 08 <Checksum> 00’`
	- <Cryptogram> contains eMRTD Application identifier ‘A0 00 00 02 47 10 01’ encrypted according to the SM being used

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	4. The eMRTD MUST return:
	`’7C <L7C> 84 <L84> <Ephemeral Public Key> 90 00’`
	5. The eMRTD MUST return:
	`’7C <L7C> 86 <L86> <Authentication Token> 90 00’`
	6. The eMRTD MUST return status bytes ’90 00’ in a valid SM response.

3.9.44 Test case ISO7816_P_44

Purpose	General Authenticate APDU to map the nonce - test borderline cases for x- and y-coordinates (large x coordinate)
Version	2.0
References	[R7] TR-SAC
Profile	PACE, PACE-ECDH, PACE-GM
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	- Use a ephemeral public key with an x-coordinate having its highest bit set to 1
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`’00 86 00 00 <Lc> 7C <L7C> 85 <L85> <Authentication Token> <Le>’`
	6. To verify the chip's ability to start the SM with the session keys, the Select LDS command is sent to the chip under SM.
	`‘0C A4 04 0C <Lc> 87 <L87> 01 <Cryptogram> 8E 08 <Checksum> 00’`
	- <Cryptogram> contains eMRTD Application identifier ‘A0 00 00 02 47 10 01’ encrypted according to the SM being used

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	4. The eMRTD MUST return:
	`’7C <L7C> 84 <L84> <Ephemeral Public Key> 90 00’`
	5. The eMRTD MUST return:
	`’7C <L7C> 86 <L86> <Authentication Token> 90 00’`
	6. The eMRTD MUST return status bytes ’90 00’ in a valid SM response.

3.9.45 Test case ISO7816_P_45

Purpose	General Authenticate APDU to map the nonce - test borderline cases for x- and y-coordinates (small y coordinate)

Version	2.0
References	[R7] TR-SAC
Profile	PACE, PACE-ECDH, PACE-GM
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly
Test scenario	1. Send the given MSE: Set AT APDU with MRZ password: ’00 22 C1 A4 <Lc> 80 <L$_{80}$> <PACE OID> 83 01 01 84 <L$_{84}$> <private key reference>’ - <PACE OID> : valid Object Identifier to the PACE protocol - The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous. 2. Send the given General Authenticate APDU to get the encrypted nonce: ‘10 86 00 00 <Lc> 7C 00 <Le>’ 3. Send the given General Authenticate APDU to map the nonce: ‘10 86 00 00 <Lc> 7C <L$_{7C}$> 81 <L$_{81}$> <Mapping Data> <Le>’ - Use an ephemeral public key with an y-coordinate requiring at least one byte less than the fewest bytes that can represent [$\log_{256}$ q]. Pad with zero bytes. (For details on q see [R8]) 4. Send the given General Authenticate APDU to perform key agreement: ‘10 86 00 00 <Lc> 7C <L$_{7C}$> 83 <L$_{83}$> <Ephemeral Public Key> <Le>’ 5. Send the given General Authenticate APDU to perform mutual authenticate: ’00 86 00 00 <Lc> 7C <L$_{7C}$> 85 <L$_{85}$> <Authentication Token> <Le>’ 6. To verify the chip's ability to start the SM with the session keys, the Select LDS command is sent to the chip under SM. ‘0C A4 04 0C <Lc> 87 <L$_{87}$> 01 <Cryptogram> 8E 08 <Checksum> 00’ - <Cryptogram> contains eMRTD Application identifier ‘A0 00 00 02 47 10 01’ encrypted according to the SM being used
Expected results	1. The eMRTD MUST return status bytes ’90 00’ 2. The eMRTD MUST return: ’7C <L$_{7C}$> 80 <L$_{80}$> <encrypted nonce> 90 00’ 3. The eMRTD MUST return: ’7C <L$_{7C}$> 82 <L$_{82}$> <Mapping Data> 90 00’ 4. The eMRTD MUST return: ’7C <L$_{7C}$> 84 <L$_{84}$> <Ephemeral Public Key> 90 00’ 5. The eMRTD MUST return: ’7C <L$_{7C}$> 86 <L$_{86}$> <Authentication Token> 90 00’ 6. The eMRTD MUST return status bytes ’90 00’ in a valid SM response.

3.9.46 Test case ISO7816_P_46

Purpose	General Authenticate APDU to map the nonce - test borderline cases for x- and y-coordinates (large y coordinate)
Version	2.0
References	[R7] TR-SAC

Profile	PACE, PACE-ECDH, PACE-GM
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L`$_{80}$`> <PACE OID> 83 01 01 84 <L`$_{84}$`> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L`$_{7C}$`> 81 <L`$_{81}$`> <Mapping Data> <Le>’`
	- Use a ephemeral public key with an y-coordinate having its highest bit set to 1
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L`$_{7C}$`> 83 <L`$_{83}$`> <Ephemeral Public Key> <Le>’`
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`’00 86 00 00 <Lc> 7C <L`$_{7C}$`> 85 <L`$_{85}$`> <Authentication Token> <Le>’`
	6. To verify the chip's ability to start the SM with the session keys, the Select LDS command is sent to the chip under SM.
	`‘0C A4 04 0C <Lc> 87 <L`$_{87}$`> 01 <Cryptogram> 8E 08 <Checksum> 00’`
	- <Cryptogram> contains eMRTD Application identifier ‘A0 00 00 02 47 10 01’ encrypted according to the SM being used

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L`$_{7C}$`> 80 <L`$_{80}$`> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L`$_{7C}$`> 82 <L`$_{82}$`> <Mapping Data> 90 00’`
	4. The eMRTD MUST return:
	`’7C <L`$_{7C}$`> 84 <L`$_{84}$`> <Ephemeral Public Key> 90 00’`
	5. The eMRTD MUST return:
	`’7C <L`$_{7C}$`> 86 <L`$_{86}$`> <Authentication Token> 90 00’`
	6. The eMRTD MUST return status bytes ’90 00’ in a valid SM response.

3.9.47 Test case ISO7816_P_47

Purpose	General Authenticate APDU to map the nonce – value higher than the prime
Version	2.04
References	[R7] TR-SAC
Profile	PACE, PACE-DH, PACE-GM
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password: `'00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>'` - <PACE OID> : valid Object Identifier to the PACE protocol - The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous. 2. Send the given General Authenticate APDU to get the encrypted nonce: `'10 86 00 00 <Lc> 7C 00 <Le>'` 3. Send the given General Authenticate APDU to map the nonce: `'10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>'` - Use an ephemeral public key with a wrong value (value larger than the Prime) ephemeral public key = prime p + 1 4. Select LDS application 5. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip. `'<Unsecured command as defined in table 1>'`
Expected results	1. The eMRTD MUST return status bytes '90 00' 2. The eMRTD MUST return: `'7C <L7C> 80 <L80> <encrypted nonce> 90 00'` 3. The eMRTD MUST return an ISO_Checking_Error or ISO execution error 4. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 5. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.48 Test case ISO7816_P_48

Purpose	General Authenticate APDU to map the nonce – ephemeral mapping public key value is set to '00..00'
Version	2.04
References	[R7] TR-SAC
Profile	PACE, PACE-DH, PACE-GM
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly
Test scenario	1. Send the given MSE: Set AT APDU with MRZ password: `'00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>'` - <PACE OID> : valid Object Identifier to the PACE protocol - The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous. 2. Send the given General Authenticate APDU to get the encrypted nonce: `'10 86 00 00 <Lc> 7C 00 <Le>'` 3. Send the given General Authenticate APDU to map the nonce:

	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	- Use an ephemeral public key with a value set to to zero (’00..00’ over the prime length)
	4. Select LDS application
	5. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’ 2. The eMRTD MUST return: `’7C <L7C> 80 <L80> <encrypted nonce> 90 00’` 3. The eMRTD MUST return an ISO_Checking_Error or ISO execution error 4. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 5. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.49 Test case ISO7816_P_49

Purpose	General Authenticate APDU to map the nonce – wrong point (value does not belong to the curve)
Version	2.04
References	[R7] TR-SAC
Profile	PACE, PACE-ECDH, PACE-GM
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`'00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>'`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`'10 86 00 00 <Lc> 7C 00 <Le>'`
	3. Send the given General Authenticate APDU to map the nonce:
	`'10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>'`
	- Use an ephemeral public key with a wrong point (value does not belong to the curve). An arbitrary generator can be used to generate the key pair..
	4. Send the given General Authenticate APDU to perform key agreement:
	`'10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>'`
	5. Select LDS application
	6. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`'<Unsecured command as defined in table 1>'`

Expected results	1. The eMRTD MUST return status bytes '90 00' 2. The eMRTD MUST return: `'7C <L7C> 80 <L80> <encrypted nonce> 90 00'` 3. The eMRTD MUST return status bytes '90 00' or an ISO_Checking_Error or ISO execution error. Remaining steps are skipped in case of error. 4. The eMRTD MUST return an ISO_Checking_Error or ISO execution error 5. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error . 6. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.50 Test case ISO7816_P_50

Purpose	General Authenticate APDU to map the nonce – wrong point encoding (the first byte of the public key is different from '01', '02', '03' and '04')

Version	2.04
References	[R7] TR-SAC
Profile	PACE, PACE-ECDH, PACE-GM
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	- Send the ephemeral public encoded as follows : ‘77||x||y’
	4. Select LDS application
	5. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’ 2. The eMRTD MUST return: `’7C <L7C> 80 <L80> <encrypted nonce> 90 00’` 3. The eMRTD MUST return an ISO_Checking_Error or ISO execution error 4. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 5. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.51 Test case ISO7816_P_51

Purpose	General Authenticate APDU to map the nonce with invalid length for the additional nonce (size different from the expected one)
Version	2.04
References	[R7] TR-SAC
Profile	PACE, PACE-IM
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`'00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>'`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`'10 86 00 00 <Lc> 7C 00 <Le>'`
	3. Send the given General Authenticate APDU to map the nonce:
	`'10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>'`
	- The nonce sent has a wrong length (different from the expected length) : length = length + 1 (byte 0x00 is added to the start of the byte string representing the nonce).
	4. Select LDS application
	5. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`'<Unsecured command as defined in table 1>'`

Expected results	1. The eMRTD MUST return status bytes '90 00' 2. The eMRTD MUST return: `'7C <L7C> 80 <L80> <encrypted nonce> 90 00'` 3. The eMRTD MUST return an ISO_Checking_Error or ISO execution error 4. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 5. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.52 Test case ISO7816_P_52

Purpose	General Authenticate APDU to perform key agreement - wrong point encoding (the first byte of the public key is different from ‘01’, ‘02’, ‘03’ and ‘04’)
Version	2.04
References	[R7] TR-SAC
Profile	PACE, PACE-ECDH
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`'00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>'`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>'`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>'`
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>'`
	- Send the ephemeral public encoded as follows : ‘77||x||y’
	5. Select LDS application
	6. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>'`

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`'7C <L7C> 80 <L80> <encrypted nonce> 90 00'`
	3. The eMRTD MUST return:
	`'7C <L7C> 82 <L82> <Mapping Data> 90 00'`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return an ISO_Checking_Error or ISO execution error
	5. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error.
	6. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.53 Test case ISO7816_P_53

Purpose	General Authenticate APDU to perform key agreement – ephemeral public key value is set to ’00..00’
Version	2.04
References	[R7] TR-SAC
Profile	PACE, PACE-DH
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	- Use an ephemeral public key with a value set to zero (’00..00’ over the prime length)
	5. Select LDS application
	6. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return an ISO_Checking_Error or ISO execution error
	5. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error.
	6. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.54 Test case ISO7816_P_54

Purpose	General Authenticate to get the encrypted nonce command while the CLASS byte does not indicate command chaining
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`'00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>'`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`'00 86 00 00 <Lc> 7C 00 <Le>'`
	3. Select LDS application
	4. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`'<Unsecured command as defined in table 1>'`

Expected results	1. The eMRTD MUST return '90 00' 2. The eMRTD MUST return an ISO_Checking_Error or ISO execution error 3. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MAY return status bytes '69 82'. Next step is skipped in case of returning '69 82'. 4. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.55 Test case ISO7816_P_55

Purpose	General Authenticate APDU to map the nonce while the CLASS byte does not indicate command chaining
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`’00 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Select LDS application
	5. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’ 2. The eMRTD MUST return: `’7C <L7C> 80 <L80> <encrypted nonce> 90 00’` 3. The eMRTD MUST return an ISO_Checking_Error or ISO execution error 4. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 5. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.56 Test case ISO7816_P_56

Purpose	General Authenticate APDU to perform key agreement while the CLASS byte does not indicate command chaining
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`'00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>'`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`'10 86 00 00 <Lc> 7C 00 <Le>'`
	3. Send the given General Authenticate APDU to map the nonce:
	`'10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>'`
	4. Send the given General Authenticate APDU to perform key agreement:
	`'00 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>'`
	5. Select LDS application
	6. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`'<Unsecured command as defined in table 1>'`

Expected results	1. The eMRTD MUST return status bytes '90 00'
	2. The eMRTD MUST return:
	`'7C <L7C> 80 <L80> <encrypted nonce> 90 00'`
	3. The eMRTD MUST return:
	`'7C <L7C> 82 <L82> <Mapping Data> 90 00'`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to '00' and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return an ISO_Checking_Error or ISO execution error
	5. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error.
	6. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.57 Test case ISO7816_P_57

Purpose	General Authenticate APDU to perform Mutual Authenticate while the CLASS byte indicates command chaining
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`’10 86 00 00 <Lc> 7C <L7C> 85 <L85> <Authentication Token> <Le>’`
	6. To verify the chip's ability to reject Secured APDU after performing incomplete PACE v2 protocol, an arbitrary secured APDU is sent to the chip.
	`‘0C A4 04 0C <Lc> 87 <L87> 01 <Cryptogram> 8E 08 <Checksum> 00’`
	- <Cryptogram> contains eMRTD Application identifier ‘A0 00 00 02 47 10 01’ encrypted according to the SM being used

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return:
	`’7C <L7C> 84 <L84> <Ephemeral Public Key> 90 00’`
	5. The eMRTD MUST return status bytes ’68 83’
	6. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response

3.9.58 Test case ISO7816_P_58

Purpose	General Authenticate APDU to perform Mutual Authenticate is not sent and replaced by another command
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`'00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>'`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`'10 86 00 00 <Lc> 7C 00 <Le>'`
	3. Send the given General Authenticate APDU to map the nonce:
	`'10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>'`
	4. Send the given General Authenticate APDU to perform key agreement:
	`'10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>'`
	5. Replace the last command of the PACE protocol by the READ BINARY command:
	`'00 B0 9C 00 01'`
	6. To verify the chip's ability to reject Secured APDU after performing incomplete PACE v2 protocol, an arbitrary secured APDU is sent to the chip.
	`'0C A4 04 0C <Lc> 87 <L87> 01 <Cryptogram> 8E 08 <Checksum> 00'`
	- <Cryptogram> contains eMRTD Application identifier 'A0 00 00 02 47 10 01' encrypted according to the SM being used

Expected results	1. The eMRTD MUST return status bytes '90 00'
	2. The eMRTD MUST return:
	`'7C <L7C> 80 <L80> <encrypted nonce> 90 00'`
	3. The eMRTD MUST return:
	`'7C <L7C> 82 <L82> <Mapping Data> 90 00'`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to '00' and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return
	`'7C <L7C> 84 <L84> < Ephemeral Public Key > 90 00'`
	5. The eMRTD MUST return status bytes '90 00' or an ISO checking error or ISO execution error
	6. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response

3.9.59 Test case ISO7816_P_59

Purpose	General Authenticate APDU to map the nonce while an unexpected command was executed between the nonce generation and the mapping
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the READ BINARY command:
	`‘00 B0 9C 00 01’`
	4. Send the given General Authenticate APDU to map the nonce:
	`’10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	5. Select LDS application
	6. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’ 2. The eMRTD MUST return: `’7C <L7C> 80 <L80> <encrypted nonce> 90 00’` 3. The eMRTD MUST return status bytes ’90 00’ or an ISO_Checking_Error or ISO execution error 4. The eMRTD MUST return an ISO_Checking_Error or ISO execution error 5. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 6. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.60 Test case ISO7816_P_60

Purpose	General Authenticate APDU to perform the key agreement while an unexpected command was executed between the mapping and the key agreement
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`'00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>'`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`'10 86 00 00 <Lc> 7C 00 <Le>'`
	3. Send the given General Authenticate APDU to map the nonce:
	`'10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>'`
	4. Send the READ BINARY command:
	`'00 B0 9C 00 01'`
	5. Send the given General Authenticate APDU to perform key agreement:
	`'10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>'`
	6. Send the given General Authenticate APDU to perform mutual authenticate:
	`'00 86 00 00 <Lc> 7C <L7C> 85 <L85> <Authentication Token> <Le>'`
	- <Authentication Token> can be any valid dummy data.
	7. Select LDS application
	8. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`'<Unsecured command as defined in table 1>'`

Expected results	1. The eMRTD MUST return status bytes '90 00'
	2. The eMRTD MUST return:
	`'7C <L7C> 80 <L80> <encrypted nonce> 90 00'`
	3. The eMRTD MUST return:
	`'7C <L7C> 82 <L82> <Mapping Data> 90 00'`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to '00' and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return '90 00' or an ISO_Checking_Error or ISO execution error
	5. The eMRTD MUST return an ISO_Checking_Error or ISO execution error
	6. The eMRTD MUST return an ISO_Checking_Error or ISO execution error
	7. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'.

	eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 8. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.61 Test case ISO7816_P_61

Purpose	General Authenticate APDU to perform the mutual authentication while an unexpected command was executed between the key agreement and the mutual authentication
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly
Test scenario	1. Send the given MSE: Set AT APDU with MRZ password: `’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’` - <PACE OID> : valid Object Identifier to the PACE protocol - The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous. 2. Send the given General Authenticate APDU to get the encrypted nonce: `‘10 86 00 00 <Lc> 7C 00 <Le>’` 3. Send the given General Authenticate APDU to map the nonce: `’10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’` 4. Send the given General Authenticate APDU to perform key agreement: `‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’` 5. Send the READ BINARY command: `‘00 B0 9C 00 01’` 6. Send the given General Authenticate APDU to perform mutual authenticate: `’00 86 00 00 <Lc> 7C <L7C> 85 <L85> <Authentication Token> <Le>’` 7. Select LDS application 8. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip. `‘<Unsecured command as defined in table 1>’`
Expected results	1. The eMRTD MUST return status bytes ’90 00’ 2. The eMRTD MUST return: `’7C <L7C> 80 <L80> <encrypted nonce> 90 00’` 3. The eMRTD MUST return: `’7C <L7C> 82 <L82> <Mapping Data> 90 00’` *Note:* *In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.* 4. The eMRTD MUST return: `’7C <L7C> 84 <L84> <Ephemeral Public Key> 90 00’`

	5. The eMRTD MUST return ’90 00’ or an ISO checking error or ISO execution error 6. The eMRTD MUST return an ISO_Checking_Error or ISO execution error 7. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 8. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.62 Test case ISO7816_P_62

Purpose	General Authenticate APDU to perform Mutual Authenticate without using Tag 7F49 in the input for the authentication token
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`’10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`’00 86 00 00 <Lc> 7C <L7C> 85 <L85> <Authentication Token> <Le>’`
	`Use Tag 30 instead of Tag 7F49`
	6. Select LDS application
	7. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	Note:

	In case of Integrated Mapping, <L82> MUST be set to '00' and < Mapping Data> MUST be empty. 4. The eMRTD MUST return: `'7C <L7C> 84 <L84> <Ephemeral Public Key> 90 00'` 5. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning 6. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 7. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.63 Test case ISO7816_P_63

Removed in version v2.0

3.9.64 Test case ISO7816_P_64

Purpose	MSE: Set AT command without data object 80
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`'00 22 C1 A4 <Lc>  83 01 01 84 <L84> <private key reference>'`
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`'10 86 00 00 <Lc> 7C 00 <Le>'`
	3. Send the given General Authenticate APDU to map the nonce:
	`'10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>'`
	4. Send the given General Authenticate APDU to perform key agreement:
	`'10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>'`
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`'00 86 00 00 <Lc> 7C <L7C> 85 <L85> <Authentication Token> <Le>'`
	6. Select LDS application
	7. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`'<Unsecured command as defined in table 1>'`

Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error or '90 00' 2. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning or '90 00'. Steps 3 to 5 are skipped in case of ISO checking error or ISO execution error or ISO Warning.

	3. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning or ‘90 00’. Steps 4 and 5 are skipped in case of ISO checking error or ISO execution error or ISO Warning. 4. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning or ‘90 00’. Step 5 is skipped in case of ISO checking error or ISO execution error or ISO Warning. 5. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning. 6. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error **.** 7. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.65 Test case ISO7816_P_65

Purpose	MSE: Set AT command with an empty data object 80
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc>  80 00 83 01 01 84 <L84> <private key reference>’`
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Select LDS application
	4. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error or ’90 00’ 2. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning 3. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 4. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.66 Test case ISO7816_P_66

Purpose	MSE: Set AT command with a too long data object 80.
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`'00 22 C1 A4 <Lc>  80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>'`
	- <PACE OID> : invalid PACE OID which is too long (e.g.: `'04 00 7F 00 07 02 02 04 02 02 02')`
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`'10 86 00 00 <Lc> 7C 00 <Le>'`
	3. Select LDS application
	4. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`'<Unsecured command as defined in table 1>'`

Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error or ’90 00’ 2. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning 3. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 4. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.67 Test case ISO7816_P_67

Purpose	MSE: Set AT command with a too short data object 80.
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`‘00 22 C1 A4 <Lc>  80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : invalid PACE OID which is too short (e.g.: `’04 00 7F 00 07 02 02 04 02’)`
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Select LDS application
	4. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error or ’90 00’ 2. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning 3. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 4. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.68 Test case ISO7816_P_68

Purpose	MSE: Set AT command with a given PACE OID not declared in the EF.CardAccess (not supported)
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid PACE OID which is not supported by the eMRTD
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Select LDS application
	4. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error or ’90 00’ 2. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning 3. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 4. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.69 Test case ISO7816_P_69

Purpose	MSE: Set AT command without data object 83
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`'00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 84 <L84> <private key reference>'`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`'10 86 00 00 <Lc> 7C 00 <Le>'`
	3. Select LDS application
	4. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`'<Unsecured command as defined in table 1>'`

Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error or '90 00' 2. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning 3. eMRTD supporting BAC and PACEv2 MUST return status bytes '90 00'. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes '90 00' instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 4. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.70 Test case ISO7816_P_70

Purpose	MSE: Set AT command with an empty data object 83
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 00 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Select LDS application
	4. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error or ’90 00’ 2. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning 3. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 4. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.71 Test case ISO7816_P_71

Purpose	MSE: Set AT command with an incorrect length byte for password reference (DO 83)
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly
Test scenario	1. Send the given MSE: Set AT APDU with MRZ password: `’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 03 01 84 <L84> <private key reference>’` - <PACE OID> : valid Object Identifier to the PACE protocol - The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous. 2. Send the given General Authenticate APDU to get the encrypted nonce: `‘10 86 00 00 <Lc> 7C 00 <Le>’` 3. Select LDS application 4. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip. `‘<Unsecured command as defined in table 1>’`
Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error or ’90 00’ 2. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning 3. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 4. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.72 Test case ISO7816_P_72

Purpose	MSE: Set AT command with a too long password reference (DO 83)
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 02 01 FF 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Select LDS application
	4. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error or ’90 00’ 2. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning 3. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 4. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.73 Test case ISO7816_P_73

Purpose	Positive test using domain parameter reference (DO 84) but eMRTD supports only one set of domain parameters
Version	2.0
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. The PACEv2 parameters were successfully read from EF.CardAccess. 3. EF.CardAccess contains one or more PACEInfo entries having all the same parameter Id.

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`'00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>'`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU even if the domain parameters are not ambiguous,
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`'10 86 00 00 <Lc> 7C 00 <Le>'`
	3. Send the given General Authenticate APDU to map the nonce:
	`'10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>'`
	4. Send the given General Authenticate APDU to perform key agreement:
	`'10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>'`
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`'00 86 00 00 <Lc> 7C <L7C> 85 <L85> <Authentication Token> <Le>'`
	6. To verify the chip's ability to start the SM with the session keys, the Select LDS command is sent to the chip under SM.
	`'0C A4 04 0C <Lc> 87 <L87> 01 <Cryptogram> 8E 08 <Checksum> 00'`
	- <Cryptogram> contains eMRTD Application identifier 'A0 00 00 02 47 10 01' encrypted according to the SM being used

Expected results	1. The eMRTD MUST return status bytes '90 00'
	2. The eMRTD MUST return:
	`'7C <L7C> 80 <L80> <encrypted nonce> 90 00'`
	3. The eMRTD MUST return:
	`'7C <L7C> 82 <L82> <Mapping Data> 90 00'`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to '00' and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return:
	`'7C <L7C> 84 <L84> <Ephemeral Public Key> 90 00'`
	5. The eMRTD MUST return:
	`'7C <L7C> 86 <L86> <Authentication Token> 90 00'`
	6. The eMRTD MUST return status bytes '90 00' in a valid SM response.

3.9.74 Test case ISO7816_P_74

Purpose	Positive test without domain parameter reference (DO 84) and eMRTD supports only one set of domain parameters
Version	2.07
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. The PACEv2 parameters were successfully read from EF.CardAccess. 3. EF.CardAccess contains only one PACEInfo or more than one PACEInfo which are not ambiguous (only one parameter Id or distinct PACE OID).

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`‘00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST not be included in the APDU
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`’00 86 00 00 <Lc> 7C <L7C> 85 <L85> <Authentication Token> <Le>’`
	6. To verify the chip's ability to start the SM with the session keys, the Select LDS command is sent to the chip under SM.
	`‘0C A4 04 0C <Lc> 87 <L87> 01 <Cryptogram> 8E 08 <Checksum> 00’`
	- <Cryptogram> contains eMRTD Application identifier ‘A0 00 00 02 47 10 01’ encrypted according to the SM being used

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return:
	`’7C <L7C> 84 <L84> <Ephemeral Public Key> 90 00’`
	5. The eMRTD MUST return:
	`’7C <L7C> 86 <L86> <Authentication Token> 90 00’`
	6. The eMRTD MUST return status bytes ’90 00’ in a valid SM response.

3.9.75 Test case ISO7816_P_75

Purpose	MSE: Set AT command with an empty domain parameter reference (DO 84)
Version	2.05
References	[R7] TR-SAC
Profile	PACE
Preconditions	This test can be performed in case 2 or more parameterId values have been assigned to the same PACE OID in the PACEInfo entries in EF.CardAccess. 1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`‘00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 00’`
	- <PACE OID> : Valid Object Identifier for the PACE protocol that has been assigned to 2 or more ParameterId values in EF.CardAccess
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Select LDS application
	4. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return an ISO checking error or ISO execution error or ’90 00’ 2. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning 3. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 4. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.9.76 Test case ISO7816_P_76

Purpose	General Authenticate APDU to perform Mutual Authenticate with a longer authentication token
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 80 <L80> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Send the given General Authenticate APDU to map the nonce:
	`‘10 86 00 00 <Lc> 7C <L7C> 81 <L81> <Mapping Data> <Le>’`
	4. Send the given General Authenticate APDU to perform key agreement:
	`‘10 86 00 00 <Lc> 7C <L7C> 83 <L83> <Ephemeral Public Key> <Le>’`
	5. Send the given General Authenticate APDU to perform mutual authenticate:
	`’00 86 00 00 <Lc> 7C <L7C> 85 <L85> <Authentication Token> <Le>’`
	- Extend the value of the Authentication token by one byte. <L85> must be correctly computed
	7. To verify the chip's ability to reject Secured APDU after performing incomplete PACE v2 protocol, an arbitrary secured APDU is sent to the chip.
	`‘0C A4 04 0C <Lc> 87 <L87> 01 <Cryptogram> 8E 08 <Checksum> 00’`
	- <Cryptogram> contains eMRTD Application identifier ‘A0 00 00 02 47 10 01’ encrypted according to the SM being used

Expected results	1. The eMRTD MUST return status bytes ’90 00’
	2. The eMRTD MUST return:
	`’7C <L7C> 80 <L80> <encrypted nonce> 90 00’`
	3. The eMRTD MUST return:
	`’7C <L7C> 82 <L82> <Mapping Data> 90 00’`
	Note:
	In case of Integrated Mapping, <L82> MUST be set to ‘00’ and < Mapping Data> MUST be empty.
	4. The eMRTD MUST return:
	`’7C <L7C> 84 <L84> <Ephemeral Public Key> 90 00’`
	5. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning
	6. The eMRTD MUST return an ISO checking error or ISO execution error

	in an unsecured response

3.9.77 Test case ISO7816_P_77

Purpose	MSE: Set AT command with a PACE OID with tag ‘0x06’instead of Tag '0x80'
Version	2.04
References	[R7] TR-SAC
Profile	PACE
Preconditions	1. Reset the chip 2. EF CardAccess has been read correctly

Test scenario	1. Send the given MSE: Set AT APDU with MRZ password:
	`’00 22 C1 A4 <Lc> 06 <LPACE OID> <PACE OID> 83 01 01 84 <L84> <private key reference>’`
	- <PACE OID> : valid Object Identifier to the PACE protocol
	- The private key reference MUST be included in the APDU if and only if the domain parameters are ambiguous.
	2. Send the given General Authenticate APDU to get the encrypted nonce:
	`‘10 86 00 00 <Lc> 7C 00 <Le>’`
	3. Select LDS application
	4. To verify the chip's ability to still require Secured APDU after performing incomplete PACE v2 protocol, an arbitrary unsecured SM-requiring APDU is sent to the chip.
	`‘<Unsecured command as defined in table 1>’`

Expected results	1. The eMRTD MUST return an ISO_Checking_Error or ISO execution error 2. The eMRTD MUST return an ISO checking error or ISO execution error or ISO_Warning 3. eMRTD supporting BAC and PACEv2 MUST return status bytes ’90 00’. eMRTD supporting only PACEv2 MUST return an ISO checking error or ISO execution error or status bytes ’90 00’ instead. Next step is skipped in case of returning ISO_Checking_Error or ISO execution error. 4. The eMRTD MUST return an ISO checking error or ISO execution error in an unsecured response.

3.10 Unit ISO7816_Q – Select and Read EF CardAccess

This test unit contains all mandatory tests regarding the Select and Read binary command applied to EF CardAccess. EF CardAccess MUST be a transparent elementary file contained in the master file.

3.10.1 Test case ISO7816_Q_01

Purpose	Accessing EF.CardAccess with explicit file selection and Read Binary
Version	2.0
References	[R7] TR-SAC §3.1
Profile	PACE
Preconditions	1. Reset the chip
Test scenario	1. Send the following Select APDU to the eMRTD `'00 A4 02 0C 02 01 1C'` 2. Send the following Read Binary APDU to the eMRTD `'00 B0 00 00 01'`
Expected results	1. The eMRTD MUST return status bytes '90 00' 2. The eMRTD MUST return byte '31' followed by status bytes '90 00'

3.10.2 Test case ISO7816_Q_02

Purpose	Accessing EF.CardAccess with implicit file selection (ReadBinary with SFI)
Version	2.0
References	[R7] TR-SAC §3.1
Profile	PACE
Preconditions	1. Reset the chip
Test scenario	1. Send the following Read Binary APDU to the eMRTD `'00 B0 9C 00 01'`
Expected results	1. The eMRTD MUST return byte '31' followed by status bytes '90 00'

3.10.3 Test case ISO7816_Q_03

Purpose	Accessing EF.CardAccess with explicit file selection and Read Binary OddIns
Version	2.0
References	[R7] TR-SAC §3.1
Profile	PACE, OddIns
Preconditions	1. Reset the chip
Test scenario	1. Send the following Select APDU to the eMRTD `'00 A4 02 0C 02 01 1C'` 2. Send the following Read Binary APDU to the eMRTD `'00 B1 00 00 04 54 02 00 00 03'`
Expected results	1. The eMRTD MUST return status bytes '90 00' 2. The eMRTD MUST return bytes '53 L_{data} data' followed by status bytes '90 00'

3.10.4 Test case ISO7816_Q_04

Purpose	Accessing EF.CardAccess with implicit file selection (ReadBinary OddIns with SFI)
Version	2.0
References	[R7] TR-SAC §3.1
Profile	PACE, OddIns
Preconditions	1. Reset the chip
Test scenario	1. Send the following Read Binary APDU to the eMRTD `'00 B1 00 1C 04 54 02 00 00 03'`
Expected results	1. The eMRTD MUST return byte '53 L_{data} data' followed by status bytes '90 00'

3.11 Unit ISO7816_R – Active Authentication

This test unit contains all mandatory tests regarding execution of the Active Authentication security mechanism.

3.11.1 Test Case ISO7816_R_01

Purpose	Verify the behaviour of an unprotected eMRTD in response to the INTERNAL AUTHENTICATE command (positive test)
Version	2.0
References	[R1] 9303-3 vol 2 – section IV - 8 [R4] Supplement to 9303 – appendix F
Profile	AA, Plain
Preconditions	1. Reset the chip 2. The LDS application MUST have been selected
Test scenario	1. Send the given INTERNAL AUTHENTICATE command to the eMRTD: `’00 88 00 00 08 55 66 77 88 11 22 33 44 00’.`
Expected results	1. Response data and ’90 00’ (without SM)

3.11.2 Test Case ISO7816_R_02

Purpose	Verify the behaviour of a BAC-protected eMRTD in response to the INTERNAL AUTHENTICATE command (positive test)
Version	2.0
References	[R1] 9303-3 vol 2 – section IV - 8 [R4] Supplement to 9303 – appendix F
Profile	AA, BAC
Preconditions	1. Reset the chip 2. The LDS application MUST have been selected. 3. The BAC mechanism MUST have been performed.
Test scenario	1. Send the given INTERNAL AUTHENTICATE valid SM command: `‘0C 88 00 00 <Lc> 87 <L87> 01 <cryptogram> 97 01 00 8E 08 <checksum> 00’` - <cryptogram> contains the SM encryption of the following data : ’55 66 77 88 11 22 33 44’
Expected results	1. Response data and ’90 00’ in a valid SM response APDU.

3.11.3 Test Case ISO7816_R_03

Purpose	Verify the behaviour of an eMRTD in response to the INTERNAL AUTHENTICATE command when RND.IFD < 8 bytes
Version	2.04
References	[R1] 9303-3 vol 2 – section IV - 8 [R4] Supplement to 9303 – appendix F
Profile	AA
Preconditions	1. Reset the chip 2. The LDS application MUST have been selected. 3. If any access control mechanism is supported, this mechanism MUST have been performed.
Test scenario	1. Send the INTERNAL AUTHENTICATE valid SM command with RND.IFD ’11 22 33 44’
Expected results	1. ISO checking error or ISO execution error . If any access control mechanism is supported, the response APDU MUST be encoded in a valid SM format

3.11.4 Test Case ISO7816_R_04

Purpose	Verify the behaviour of an eMRTD in response to the INTERNAL AUTHENTICATE command when RND.IFD > 8 bytes
Version	2.04
References	[R1] 9303-3 vol 2 – section IV - 8 [R4] Supplement to 9303 – appendix F
Profile	AA
Preconditions	1. Reset the chip 2. The LDS application MUST have been selected. 3. If any access control mechanism is supported, this mechanism MUST have been performed.
Test scenario	1. Send the INTERNAL AUTHENTICATE valid SM command with RND.IFD ’11 22 33 44 55 66 77 88 99’
Expected results	1. ISO checking error or ISO execution error . If any access control mechanism is supported, the response APDU MUST be encoded in a valid SM format

3.11.5 Test Case ISO7816_R_05

Purpose	This test checks the RSA signature that is generated during Active Authentication.
Version	2.0
References	[R1] 9303-3 vol 2 – section IV - 8 [R4] ISO/IEC 9796-2 RFC-3280 RFC-3279
Profile	AA, AA-RSA
Preconditions	1. EF.DG15 has been retrieved from the eMRTD 2. EF.DG15 contains a valid RSA public key 3. The RND.IFD and the signature that has been generated by the eMRTD are available
Test scenario	1. Obtain the plaintext signature from the Internal Authenticate response. 2. Decipher the Active Authentication signature using the Public Key from EF.DG15. 3. "Signature Opening" - Check the leftmost 2 bits of the Recoverable String. 4. "Signature Opening" - Check the trailer of the Recoverable String. 5. "Intermediate String Recovery" - Retrieve the number of padding bits from the beginning of the Recoverable String. 6. "Trailer Recovery" - Check the last byte of the Recoverable String. 7. "Hash Code Checking" - Retrieve the hash code from the Recoverable String
Expected results	1. The length of the signature MUST be in accordance with the length of the public key from EF.DG15 2. The length of the deciphered signature MUST be in accordance with the length of the public key from EF.DG15 3. The leftmost 2 bits of the Recoverable String MUST be equal to '01'b. 4. The rightmost 4 bits of the Recoverable String MUST be equal to '1100'b. 5. The number of padding bits equal to '0'b following the 3rd bit of the Recoverable String MUST be less than 8. 6. The trailer of the Recoverable String MUST be 'BC' if option 1 is used with SHA-1 ‘38CC’ if option 2 is used with SHA-224 ‘34CC’ if option 2 is used with SHA-256 ‘36CC’ if option 2 is used with SHA-384 ‘35CC’ if option 2 is used with SHA-512 7. . The hash code MUST match the hash calculated over M1||M2 (M1 is the nonce that has been generated by the eMRTD; M2 is RND.IFD)

3.11.6 Test Case ISO7816_R_06

Purpose	This test checks the ECDSA signature that is generated by the eMRTD during Active Authentication.
Version	2.0
References	[R1] 9303-3 vol 2 – section IV - 8 [R4] Supplement to 9303 – appendix F RFC-3280 RFC-3279
Profile	AA, AA-ECDSA
Preconditions	1. EF.DG15 has been retrieved from the eMRTD 2. EF.DG14 has been retrieved from the eMRTD 3. EF.DG15 contains a valid EC public key 4. The RND.IFD and the signature that has been generated by the eMRTD are available
Test scenario	1. Obtain the plaintext signature from the Internal Authenticate Response. 2. Verify the signature using ECDSA with the selected hash provided in DG14.
Expected results	1. The length of the signature MUST be in accordance with the length of the public key from EF.DG15 2. Signature verification MUST be successful.

4 Logical Data Structure Tests

The “logical data structure” test layer analyses the encoding of the LDS objects stored on an eMRTD. This layer contains several test units, one for each LDS object (DG 1 - 16, EF.COM and EF.SOD). Another test unit verifies the integrity and consistency of the different data structures. The tests specified in this layer can be performed using a regular eMRTD or with following input data from a different source (e.g. file). The test configuration document specifies the source of the data.

4.1 Unit Test LDS_A - Tests for the EF.COM LDS Object

This unit includes all test cases concerning the EF.COM element. The general LDS header encoding is tested as well as the referred LDS and Unicode version numbers. The consistency of the data group list with respect to the available data group objects is checked in a different test unit.

4.1.1 Test Case LDS_A_01

Purpose	This test checks the template tag; the encoded LDS element starts with.
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.COM object in binary format as read from the eMRTD.
Test scenario	1. Check the very first byte of the EF.COM element
Expected results	1. First byte MUST be ‘60’
Postconditions	None

4.1.2 Test Case LDS_A_02

Purpose	This test checks the encoding of LDS element length.
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.COM object in binary format as read from the eMRTD.
Test scenario	1. Analyze the encoding of the bytes that follow the template tag 2. Verify the length of the given LDS object
Expected results	1. The bytes that follow the template tag MUST contain a valid length encoding (According to ASN.1 encoding rules). 2. The encoded length MUST match the size of the given LDS object.
Postconditions	None

4.1.3 Test Case LDS_A_03

Purpose	This test checks the LDS version referred by the EF.COM element
Version	2.03
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.COM object in binary format as read from the eMRTD.
Test scenario	1. Search for configured tag ‘5F 01’ 2. Verify the length of the tag ‘5F 01’ 3. Verify the length of LDS version DE. 4. Verify the LDS version.
Expected results	1. Tag MUST be present. 2. The bytes that follow the tag MUST contain a valid length encoding. 3. Length MUST be 4. 4. The specified LDS version MUST be ‘30 31 30 37’ or ’30 31 30 38’
Postconditions	None

4.1.4 Test Case LDS_A_04

Purpose	This test checks the Unicode version referred by the EF.COM element
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.COM object in binary format as read from the eMRTD.
Test scenario	1. Search for configured tag ‘5F 36’ 2. Verify the length of the tag ‘5F 36’ 3. Verify the length of the Unicode version DE. 4. Verify the Unicode version.
Expected results	1. Tag MUST be present. 2. The bytes that follow the tag MUST contain a valid length encoding. 3. The length MUST be 6. 4. The specified Unicode version MUST be ‘30 34 30 30 30 30’.
Postconditions	None

4.1.5 Test Case LDS_A_05

Purpose	This test checks the Unicode version referred by the EF.COM element
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.COM object in binary format as read from the eMRTD.
Test scenario	1. Search for configured tag ‘5C’ 2. Verify the length of the tag ‘5C’ 3. Verify if mandatory data groups are present. 4. Verify the validity of present data groups.
Expected results	1. Tag MUST be present. 2. The bytes that follow the tag MUST contain a valid length encoding 3. The list MUST at least contain the tags for the mandatory data groups ‘61’, ‘75’. 4. The list MUST contain only valid data group tags as specified in [R1], i.e. ‘61’, ‘75’, ‘63’, ‘76’, ‘65’, ‘66’, ‘67’, ‘68’, ‘69’, ‘6A’, ‘6B’, ‘6C’, ‘6D’, ‘6E’, ‘6F’, ‘70’
Postconditions	None

4.2 Unit Test LDS_B - Tests for the DataGroup 1 LDS object

This unit includes all test cases concerning the DG 1 element (MRZ). The general LDS header encoding is tested as well as the format of the MRZ and the calculation of the check digits.

Unit test B uses the following definitions in accordance with [R1]:

- A denotes the set of ASCII encoded alphabetic characters {“A”, “B”, … , “Z”}
- N denotes the set of ASCII encoded numeric characters {“0”, “1”, … , “9”}
- S denotes the set of ASCII encoded special characters {“<”}

4.2.1 Test Case LDS_B_01

Purpose	This test verifies the template tag with which the encoded LDS element starts.
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.DG1 object in binary format as read from the eMRTD.
Test scenario	1. Check the very first byte of the EF.DG1 element
Expected results	1. First byte MUST be ‘61’
Postconditions	None

4.2.2 Test Case LDS_B_02

Purpose	This test verifies the encoding of LDS element length.
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.DG1 object in binary format as read from the eMRTD.
Test scenario	1. Analyze the encoding of the bytes that follow the template tag 2. Verify the length of the given LDS object
Expected results	1. The bytes that follow the template tag MUST contain a valid length encoding (According to ASN.1 encoding rules). 2. The encoded length MUST match the size of the given LDS object.
Postconditions	None

4.2.3 Test Case LDS_B_03

Purpose	This test verifies the encoding of the MRZ data object.
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.DG1 object in binary format as read from the eMRTD.
Test scenario	1. Verify the length of the tag '5F 1F' 2. Verify that the length encoding is correct. 3. Verify that the encoded length equals the remaining size of DG1.
Expected results	1. The first bytes of the LDS element data MUST be the tag for the MRZ data object. 2. The bytes that follow the MRZ data object tag MUST contain a valid length encoding (According to ASN.1 encoding rules). 3. The encoded length MUST match the remaining size of the given DG1 object.
Postconditions	None

4.2.4 Test Case LDS_B_04

Purpose	This test checks the format of the document type.
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.DG1 object in binary format as read from the eMRTD. For all documents, the format of the document type is composed of the bytes 1 and 2 of the MRZ.
Test scenario	1. Analyze the first two characters of the MRZ (document type).
Expected results	1. The document type shall be an element of A, S (as defined in [R1]) and shall match the value declared in Table 1 of clause 2.2.
Postconditions	None

4.2.5 Test Case LDS_B_05

Purpose	This test checks the format of the issuing state of the MRZ.
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.DG1 object in binary format as read from the eMRTD. For all documents, the format of the issuing state is composed of bytes 3 to 5 (inclusive) of the MRZ.
Test scenario	1. Analyze the next three characters of the MRZ (issuing state).
Expected results	1. The characters of the issuing state MUST be elements of A that MAY be followed by elements of S.
Postconditions	None

4.2.6 Test Case LDS_B_06

Purpose	This test verifies the format of the holder name of the MRZ.
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.DG1 object in binary format as read from the eMRTD. Extract the holder name of the MRZ as described below: If TD3, the format of the holder name is composed of bytes 6 to 44. If TD2, the format of the holder name is composed of bytes 6 to 36. If TD1, the format of the holder name is composed of bytes 61 to 90.
Test scenario	1. Analyze the characters of the MRZ (holder name).
Expected results	1. The characters of the holder name MUST be elements of A or S. The holder name MUST start with a character that is an element of A.
Postconditions	None

4.2.7 Test Case LDS_B_07

Purpose	This test verifies the format of the document number.
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.DG1 object in binary format as read from the eMRTD. Extract the Document Number and its check digit as described below: If TD3, the format of the document number is composed of bytes 45 to 53. The check digit is the next character (54). If TD2, the format of the document number is composed of bytes 37 to 45. The check digit is the next character (46). If the check digit is an element of S, then the remaining characters of the document number are composed of bytes 65 until an element of S is encountered (71 or before), at which point the document number ends at 2 character positions before (69 or before) and the check digit is the next character (70 or before). If TD1, the format of the document number is composed of bytes 6 to 14. The check digit is the next character (15). If the check digit is an element of S, then the remaining characters of the document number are composed of bytes 16 until an element of S is encountered (30 or before), at which point the document number ends at 2 character positions before (28 or before) and the check digit is the next character (29 or before).
Test scenario	1. Analyze the characters of the MRZ (document number). 2. Analyze the next character of the MRZ (check digit).
Expected results	1. The characters of the document number MUST be elements of A or N that MAY be followed by elements of S. 2. The document number check digit must be an element of N and MUST be correct.
Postconditions	None

4.2.8 Test Case LDS_B_08

Purpose	This test verifies the format of the nationality.

Version	2.02
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.DG1 object in binary format as read from the eMRTD. Extract the nationality as described below: If TD3, the format of the nationality is composed of bytes 55 to 57. If TD2, the format of the nationality is composed of bytes 47 to 49. If TD1, the format of the nationality is composed of bytes 46 to 48.
Test scenario	1. Analyze the three characters of the MRZ (nationality).
Expected results	1. The characters of the nationality MUST be elements of A that MAY be followed by elements of S.
Postconditions	None

4.2.9 Test Case LDS_B_09

Purpose	This test verifies the format of the date of birth.
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.DG1 object in binary format as read from the eMRTD. Extract the date of birth and its check digit as described below: If TD3, the format of the date of birth and its check digit is composed of bytes 58 to 64. If TD2, the format of the date of birth and its check digit is composed of bytes 50 to 56. If TD1, the format of the date of birth and its check digit is composed of bytes 31 to 37.
Test scenario	1. Analyze the 6 characters of the MRZ (date of birth). 2. Analyze the next character of the MRZ (check digit).
Expected results	1. The six characters MUST be elements of N or S. The format is YYMMDD, where MM MUST be an element of {01 to 12} or S; and DD MUST be an element of {01 to 31} or S. 2. The check digit of the date of birth MUST be an element of N and MUST be correct. For data elements in which the number does not occupy all available character positions, the symbol < shall be used to complete vacant positions and shall be given the value of zero for the purpose of calculating the check digit.
Postconditions	None

4.2.10 Test Case LDS_B_10

Purpose	This test verifies the format of the sex.
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.DG1 object in binary format as read from the eMRTD. Extract the format of the sex. If TD3, the format of the sex is the byte 65. If TD2, the format of the sex is byte 57. If TD1, the format of the sex is byte 38.
Test scenario	1. Analyze the character of the MRZ (sex).
Expected results	1. The character of the sex MUST be an element of {“F”, “M”, “<”}.
Postconditions	None

4.2.11 Test Case LDS_B_11

Purpose	This test verifies the format of the date of expiry.
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.DG1 object in binary format as read from the eMRTD. Extract the date of expiry and its check digit as described below: If TD3, the format of the date of expiry and its check digit is composed of bytes 66 to 72. If TD2, the format of the date of birth and its check digit is composed of bytes 58 to 64. If TD1, the format of the date of birth and its check digit is composed of bytes 39 to 45.
Test scenario	1. Analyze the 6 characters of the MRZ (date of expiry). 2. Analyze the next character of the MRZ (check digit).
Expected results	1. The six characters MUST be elements of N. The format is YYMMDD, where MM MUST be an element of {01 to 12}; and DD MUST be an element of {01 to 31}. 2. The check digit of the date of expiry MUST be an element of N and MUST be valid.
Postconditions	None

4.2.12 Test Case LDS_B_12

Purpose	This test verifies the format of the optional data.
Version	2.02
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.DG1 object in binary format as read from the eMRTD. Extract the optional data and check digit for TD2 as described below: If TD3, the format of the optional data and its check digit is composed of bytes 73 to 87. If TD2, the format of the optional data is composed of bytes 65 to 71 (no check digit for TD2). If TD1, the format of the optional data is composed of bytes 16 to 30, and from 49 to 59 (no check digit for TD1).
Test scenario	1. Analyze the characters of the MRZ (optional data). 2. Analyze the next character of the MRZ (check digit).
Expected results	1. The characters of the optional data MUST be elements of A, N or S. 2. If the optional data's check digit is not present, skip this step. If the optional data is not empty (ie partly or wholly composed of elements of A, N), the optional data's check digit MUST be element of N and MUST be correct. Else, the optional data's check digit MUST be an element of {"0" or "<"}.
Postconditions	None

4.2.13 Test Case LDS_B_13

Purpose	This test verifies the format of composite check digit.
Version	2.0
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.DG1 object in binary format as read from the eMRTD. Extract the check digit as described below: If TD3, the format of the composite check digit is the byte 88. If TD2, the format of the composite check digit is the byte 72. If TD1, the format of the composite check digit is the byte 60.
Test scenario	1. Analyze the character of the MRZ (composite check digit). If TD3, the check digit is calculated by concatenating bytes 45 to 54, 58 to 64, and 66 to 87. If TD2, the check digit is calculated by concatenating bytes 37 to 46, 50 to 56, and 58 to 71. If TD1, the check digit is calculated by concatenating bytes 6 to 37, 39 to 45, and 49 to 59.
Expected results	1. The character of the composite check digit MUST be an element of N and ~~it~~ MUST be correct.
Postconditions	None

4.3 Unit Test LDS_C - Tests for the DataGroup 2 LDS object

This unit includes all test cases concerning the DG 2 element (Face). The general LDS header encoding is tested as well as the CBEFF encoded biometric template and ISO 19794 coding [R6] of the biometric object itself. Since the CBEFF and the ISO specification allow a very high degree of freedom, this unit contains tests for the mandatory elements as specified in the LDS.

Some additional (optional) tests verify the encoding optional elements. The general rule for this optional test is that if an optional element is present, it MUST be encoded according to the corresponding specification otherwise the test fails.

4.3.1 Test Case LDS_C_01

Purpose	This test checks the template tag; the encoded DataGroup 2 element starts with.
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.DG2 object in binary format as read from the eMRTD.
Test scenario	1. Check the very first byte of the EF.DG2 element
Expected results	1. First byte MUST be ‘75’
Postconditions	None

4.3.2 Test Case LDS_C_02

Purpose	This test checks the encoding of LDS element length.
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.DG2 object in binary format as read from the eMRTD.
Test scenario	1. Analyze the encoding of the bytes that follow the template tag 2. Verify the length of the given LDS object
Expected results	1. The bytes that follow the template tag MUST contain a valid length encoding (According to ASN.1 encoding rules). 2. The encoded length MUST match the size of the given LDS object.
Postconditions	None

4.3.3 Test Case LDS_C_03

Purpose	This test checks the encoding of the Biometric Information Group Template.
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.DG2 object in binary format as read from the eMRTD.
Test scenario	1. Check the first tag in the DG 2 data. 2. Verify the length of the DG 2 data. 3. Verify that the encoded length is less than size of DG 2.
Expected results	1. Tag MUST be '7F 61'. 2. This element MUST have a valid encoded length (According to ASN.1 encoding rules). 3. The encoded length MUST NOT exceed the remaining bytes of the DG 2 data element.
Postconditions	None

4.3.4 Test Case LDS_C_04

Purpose	This test checks the encoding of the number of instances stored in the Biometric Information Group Template.
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.DG2 object in binary format as read from the eMRTD.
Test scenario	1. Check the first tag inside the group template 2. Verify the length of the "number of instances" data object. 3. Verify that the encoded length is less than rest of size of DG 2.
Expected results	1. Tag MUST be '02'. 2. This element MUST have a valid encoded length (According to ASN.1 encoding rules). 3. The number of instances MUST be 1.
Postconditions	None

4.3.5 Test Case LDS_C_05

Purpose	This test checks the encoding of the first biometric information template.
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.DG2 object in binary format as read from the eMRTD.
Test scenario	1. Check the tag of the biometric information template. 2. Verify the length of the "biometric information template" data object. 3. Verify that the encoded length is less than rest of size of DG 2.
Expected results	1. Tag MUST be '7F 60'. 2. This element MUST have a valid encoded length (According to ASN.1 encoding rules). 3. The encoded length MUST NOT exceed the remaining bytes of the DG 2 element.
Postconditions	None

4.3.6 Test Case LDS_C_06

Purpose	This test checks the encoding of the biometric header template tag.
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.DG2 object in binary format as read from the eMRTD.
Test scenario	1. Check the presence of the biometric header template tag with the configured tag. 2. Verify the length of the “biometric header template” data object. 3. Verify that the encoded length is less than rest of size of DG 2.
Expected results	1. Tag MUST be ‘A1’. 2. This element MUST have a valid encoded length (According to ASN.1 encoding rules). 3. The encoded length MUST NOT exceed the remaining bytes of the DG 2 element.
Postconditions	None

4.3.7 Test Case LDS_C_07

Purpose	This test checks the presence/encoding of the CBEFF element "format owner".
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.DG2 object in binary format as read from the eMRTD. The tested CBEFF element is part of biometric header template located in LDS_C_06.
Test scenario	1. Check the presence of the “format owner” tag. 2. Verify the length of the “format owner” data object. 3. Check the length of the “format owner” value. 4. Verify the “format owner” value.
Expected results	1. Tag MUST be ‘87’. 2. This element MUST have a valid encoded length (According to ASN.1 encoding rules). 3. The length of the value field MUST be 2 bytes. 4. The value of the format owner MUST be a registered CBEFF owner. It MUST be ‘01 01’.
Postconditions	None

4.3.8 Test Case LDS_C_08

Purpose	This test checks the presence/encoding of the CBEFF element "format type".
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.DG2 object in binary format as read from the eMRTD. The tested CBEFF element is part of biometric header template located in LDS_C_06.
Test scenario	1. Check the presence of the format type tag. 2. Verify the length of the “format type” data object. 3. Check the length of the “format type” value. 4. Verify the “format type” value.
Expected results	1. Tag MUST be ‘88’. 2. This element MUST have a valid encoded length (According to ASN.1 encoding rules). 3. The length of the value field MUST be 2 bytes. 4. The value of the format type MUST be a registered CBEFF type. It MUST be ‘00 08’.
Postconditions	None

4.3.9 Test Case LDS_C_09

Purpose	This test checks the encoding of the biometric data object tag.
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.DG2 object in binary format as read from the eMRTD. The biometric data object is part of the biometric information template tested in LDS_C_05.
Test scenario	1. Check the presence of the biometric data object tag. 2. Verify the length of the biometric data object. 3. Verify that the encoded length is less than rest of size of DG 2.
Expected results	1. Tag MUST be ‘5F 2E’ 2. This element MUST have a valid encoded length (According to ASN.1 encoding rules). 3. The encoded length MUST NOT exceed the remaining bytes of the DG 2 element.
Postconditions	None

4.3.10 Test Case LDS_C_10

Purpose	This test checks the encoding of the facial header block.
Version	1.1
References	ICAO LDS [R1], [R2] ISO 19794-5 [R6]
Profile	ICAO
Preconditions	Encoded EF.DG2 object in binary format as read from the eMRTD. The biometric data object is part of the biometric data object tested in LDS_C_09.
Test scenario	1. Check the first 4 bytes of the header block (Format identifier) 2. Check the next 4 bytes of the header block (Version number) 3. Check the record length element. 4. Check the Number of Facial Images element.
Expected results	1. The format identifier MUST be ‘46 41 43 00’. 2. The version number MUST be ‘30 31 30 00’. 3. The length MUST NOT exceed the remaining bytes of the DG2 element and MUST match the encoded length of the biometric data object. 4. The number of facial images MUST at least be 1.
Postconditions	None

4.3.11 Test Case LDS_C_11

Purpose	This test checks the encoding of the facial information block. This test is mandatory for the first facial information block and SHOULD be repeated for further optional facial images.
Version	1.1
References	ICAO LDS [R1], [R2] ISO 19794-5 [R6]
Profile	ICAO
Preconditions	Encoded EF.DG2 object in binary format as read from the eMRTD.
Test scenario	1. Check the Facial Record Data Length. 2. Check the number of facial feature points. 3. Check the gender element. 4. Check the eye colour element. 5. Check the hair colour element. 6. Check Pose Angle - Yaw. 7. Check Pose Angle - Pitch. 8. Check Pose Angle - Roll. 9. Check Pose Angle Uncertainty –Yaw. 10. Check Pose Angle Uncertainty –Pitch. 11. Check Pose Angle Uncertainty –Roll.
Expected results	1. The Facial Record Data Length MUST be at least 32 bytes and MUST NOT exceed the remaining size of the biometric data object. 2. The size of the feature point structures (8 * number of facial feature points) MUST NOT exceed the remaining size of the biometric data object 3. The gender MUST be encoded as ‘00’, ‘01’, ‘02’, or ‘FF’. 4. The eye colour MUST be encoded as ‘00’, ‘01’, ‘02’, ‘03’, ‘04’, ‘05’, ‘06’, ‘07’, or ‘FF’. 5. The hair colour MUST be encoded as ‘00’, ‘01’, ‘02’, ‘03’, ‘04’, ‘05’, ‘06’, ‘07’, or ‘FF’. 6. The Pose Angle - Yaw MUST be equal or less than 181. 7. The Pose Angle - Pitch MUST be equal or less than 181. 8. The Pose Angle - Roll MUST be equal or less than 181. 9. The Pose Angle Uncertainty - Yaw MUST be equal or less than 181. 10. The Pose Angle Uncertainty - Pitch MUST be equal or less than 181. 11. The Pose Angle Uncertainty - Roll MUST be equal or less than 181.
Postconditions	None

4.3.12 Test Case LDS_C_12

Purpose	This test checks the encoding of the facial feature points. It is conditional and applies only if there are feature points encoded. This test SHOULD be repeated for every present feature point. See LDS_C_11 for the number of feature points.
Version	1.1
References	ICAO LDS [R1], [R2] ISO 19794-5 [R6]
Profile	ICAO
Preconditions	Encoded EF.DG2 object in binary format as read from the eMRTD.
Test scenario	1. Check the feature point type.
Expected results	1. The feature point type MUST be 1.
Postconditions	None

4.3.13 Test Case LDS_C_13

Purpose	This test checks the encoding of the image information block. This test is mandatory for the first image information block and SHOULD be repeated for further optional facial images.
Version	1.1
References	ICAO LDS [R1], [R2] ISO 19794-5 [R6]
Profile	ICAO
Preconditions	Encoded EF.DG2 object in binary format as read from the eMRTD.
Test scenario	1. Check the face image type. 2. Check the image data type.
Expected results	1. The face image type MUST be encoded as ‘00’, ‘01’, or ‘02’. 2. The image data type MUST be encoded as ‘00’ or ‘01’.
Postconditions	None

4.4 Unit Test LDS_D - Tests for the SOD LDS object

This unit includes all test cases concerning the EF.SOD element. The general LDS header encoding is tested as well as the contained CMS (PKCS#7) signed content object.

In order verify the signing certificate signature the corresponding country signing certificate is needed. For the verification of the LDS security object, the binary data group objects and the EF.COM is needed as read from the eMRTD.

4.4.1 Test Case LDS_D_01

Purpose	This test checks the template tag; the encoded DataGroup 2 element starts with.
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.SOD object in binary format as read from the eMRTD.
Test scenario	1. Check the very first byte of the EF.SOD element.
Expected results	1. First byte MUST be ‘77’.
Postconditions	None

4.4.2 Test Case LDS_D_02

Purpose	This test checks the encoding of LDS element length.
Version	1.1
References	ICAO LDS [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.SOD object in binary format as read from the eMRTD.
Test scenario	1. Analyze the encoding of the bytes that follow the template tag 2. Verify the length of the given LDS object
Expected results	1. The bytes that follow the template tag MUST contain a valid length encoding (According to ASN.1 encoding rules). 2. The encoded length MUST match the size of the given LDS object.
Postconditions	None

4.4.3 Test Case LDS_D_03

Purpose	This test checks the ASN#1 encoding of a PCKS#7 signedData object.
Version	1.1
References	ICAO PKI [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.SOD object in binary format as read from the eMRTD.
Test scenario	1. Check that the element has a sound ASN.1 structure.
Expected results	1. The PKCS#7 signed data object included as the value in the LDS true template MUST be encoded according to the DER format.
Postconditions	None

4.4.4 Test Case LDS_D_04

Purpose	This test checks the value that is encoded into the signedData element.
Version	1.1
References	ICAO PKI [R1], [R2] and DOC9303
Profile	ICAO
Preconditions	Encoded EF.SOD object in binary format as read from the eMRTD.
Test scenario	1. Check the SignedData version value. 2. Check the digestAlgorithms list. 3. Check the eContentType. 4. Check the certificates list.
Expected results	1. The version number MUST be 3. 2. All OIDs MUST be valid. This list SHOULD contain all used digestAlgorithms in this signedData container. It MUST contain only digestAlgorithms specified in the PKI report: 1.3.14.3.2.26 (SHA1) 2.16.840.1.101.3.4.2.1 (SHA-2 256) 2.16.840.1.101.3.4.2.2 (SHA-2 384) 2.16.840.1.101.3.4.2.3 (SHA-2 512) 2.16.840.1.101.3.4.2.4 (SHA-2 224) 3. The eContentType MUST have OID id-icao-ldsSecurityObject, see Annex A3.2. 4. According to the PKI Report; the certificate list MAY contain the Document Signer Certificate. If this is the case, the Document Signer Certificate is tested in LDS_D_7. Other certificates SHOULD NOT be included in this list.
Postconditions	None

4.4.5 Test Case LDS_D_05

Purpose	This test checks the SignerInfo element of the signedData structure. The signedData Structure MUST at least contain one signer info. If there is more than one signer info, although this is not recommended in the PKI report, this test MUST be repeated for each element.
Version	1.1
References	ICAO PKI [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.SOD object in binary format as read from the eMRTD.
Test scenario	1. Check the signer info version value. 2. Check the choice of the sid element. 3. Check if the certificate identified in the sid is included in the signed data certificates list or available in the PKD. 4. Check the digestAlgorithm identifier. 5. Check the signedAttrs element. 6. Check the MessageDigest Attribute. 7. Check the SigningTime attribute if present. 8. Check the signatureAlgorithm element. 9. Check the signature element. It is verified with the signer certificates public key and the hash value produced over the signedAttributes.
Expected results	1. The version number MUST be 1 or 3. 2. The choice of the sid element MUST match the signer info version value. (Version 1 if issuerandSerialNumber is used and 3 if subjectKeyIdentifier is used). 3. Certificate MUST be available. 4. The digestAlgorithmID MUST be included in the algorithm list. 5. The signed attributes list MUST contain the MessageDigest attribute. 6. The value of the message digest attribute MUST match the hash value of the eContent element. (Using the digestAlgorithm specified above) 7. If there's a SigningTime attribute present, the signing time MUST be within the validity period of the signing certificate. 8. The signature algorithm MUST refer to an algorithm specified in [R2]: RSA, DSA, or ECDSA. 9. The signature MUST be valid.
Postconditions	None

4.4.6 Test Case LDS_D_06

Purpose	This test checks the LDS Security Object stored as eContent in the signedData Object. The LDS Security Object is stored as the eContent element in the signedData Structure.
Version	2.03
References	ICAO PKI [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.SOD object in binary format as read from the eMRTD. For the data group hash verification this test needs also the binary data group objects as read from the eMRTD.
Test scenario	1. Check the ASN.1 encoding of the LDS Security Object. 2. Check the security object version element. 3. Check the digestAlgorithm identifier. 4. Check the DataGroupHash Sequence. 5. Check the dataGroup numbers in the DataGroup Hash Sequence. 6. Check the dataGroup numbers in the DataGroup Hash Sequence. 7. Check the dataGroup hash values in the Hash Sequence. Compare the hash value with the corresponding data group binary objects. 8. If the security object version is 1, check LDS Version Info element. If the security object version is 0, skip this step.
Expected results	1. The object MUST be encoded according to the DER syntax. 2. The version number MUST be 0 or 1. 3. The digestAlgorithm identifier MUST be one of the algorithms specified in [R2]: SHA1, SHA-224, SHA-256, SHA-384, and SHA-512. 4. The Sequence MUST contain at least 2 entries for DG 1 and 2. 5. The Sequence MUST contain a hash value for all present data groups. There MUST be no additional hash value for non-existing data groups. 6. The referred dataGroups MUST match the DataGroup list in the EF.COM. 7. All hash values MUST be valid. 8. LDS Version Info element shall be present and shall be ‘30 31 30 38’
Postconditions	None

4.4.7 Test Case LDS_D_07

Purpose	This test checks the signing certificate used to verify the EF.SOD object. The certificate can be read from the SOD object or MUST be retrieved from the PKD.
Version	1.1
References	ICAO PKI [R1], [R2]
Profile	ICAO
Preconditions	Encoded EF.SOD object in binary format as read from the eMRTD. For the verification of the signing certificate signature, the country signing certificate is required.
Test scenario	1. Check the ASN.1 encoding of the signing certificate. 2. Check the signing certificate version element. 3. Check the signature element. 4. Check the certificates validity period element. 5. Check the certificates issuer element. 6. Check the subjectPublicKeyInfo element. 7. Check the AuthorityKeyIdentifier extension in the signing certificate. 8. Check that the SubjectKeyIdentifier extension of the country signing certificate matches the AuthorityKeyIdentifier of the signing certificate. 9. Check the keyUsage extension of the signing certificate. 10. Check the signatureAlgorithm element. 11. Verify the signatureValue of the signing certificate with the public key of the country signing certificate.
Expected results	1. The object MUST be encoded according to the DER syntax. 2. The version MUST be v3 (Value for v3 is 2). 3. The algorithm specified here MUST match the OID in the signatureAlgorithm field. 4. It MUST use UTC time until 2049 from the on GeneralisedTime. NOTE: It is not necessary that the certificate is still valid; it MUST only have been valid at signing time, which is tested in LDS_D_5. 5. The issuer MUST match the subject of the provided country signing certificate. 6. This element MUST refer to an algorithm specified in [R2]: RSA, ECDSA, or DSA. 7. This extension MUST be present and MUST contain a keyIdentifier value. 8. AuthorityKeyIdentifier MUST match the SubjectKeyIdentifier of the country signing certificate. 9. The keyUsage extension MUST be “critical” and the digitalSignature bit MUST be asserted. 10. The signatureAlgorithm element MUST be one of the algorithms specified in [R2]: RSA, ECDSA, or DSA. 11. The certificate signature MUST be valid.
Postconditions	None

4.5 Unit LDS_E – Data Group 14

This unit contains all mandatory tests regarding the coding of data group 14. DG14 contains SecurityInfo structures related to the various security protocols supported by the eMRTD.
For the PACE profile, DG14 MUST contain a copy of each of the SecurityInfos stored in EF.CardAccess.
For the AA ECDSA profile, DG14 contains ActiveAuthenticationInfo SecurityInfos.
Further SecurityInfos belonging to other protocols such as Chip Authentication may also be included in DG14 and are covered by other test cases: the existing document TR03105 part 3.2 defines test cases LDS_E_01 to LDS_E_05.

4.5.1 Test case LDS_E_06

Purpose	Test the coherency between the DG14 and EF.CardAccess
Version	2.0
References	[R7] TR-SAC §3.1
Profile	PACE
Preconditions	1. Data group 14 MUST have been read from the eMRTD
Test scenario	1. Check the SecurityInfo structures stored in the CardAccess are duplicated in the DG14.
Expected results	1. Each SecurityInfo structure stored in the CardAccess file is also present in DG14.

4.5.2 Test case LDS_E_07

Purpose	Test the ASN.1 encoding of the SecurityInfos
Version	2.0
References	[R4] Supplement to 9303 – appendix F
Profile	AA, AA-ECDSA
Preconditions	1. EF.DG14 has been retrieved from the eMRTD
Test scenario	1. Check the SecurityInfos element
Expected results	1. The SecurityInfos MUST contain an ActiveAuthenticationInfo element with PROTOCOL OID is 2.23.136.1.1.5.

4.5.3 Test case LDS_E_08

Purpose	Test the ASN.1 encoding of the ActiveAuthenticationInfo
Version	2.0
References	[R4] Supplement to 9303 – appendix F
Profile	AA, AA-ECDSA
Preconditions	1. EF.DG14 has been retrieved from the eMRTD
Test scenario	1. Check the algorithm identifier of the ActiveAuthenticationInfo 2. Check the version of the ActiveAuthenticationInfo 3. Check the signatureAlgorithm of the ActiveAuthenticationInfo
Expected results	1. The protocol identifier MUST be: id-AA: (OID : 2.23.136.1.1.5) 2. The version MUST be encoded as INTEGER and MUST be 1 3. The signatureAlgorithm MUST be one of the following : - ecdsa-plain-SHA1: (OID: 0.4.0.127.0.7.1.1.4.1.1) - ecdsa-plain-SHA224: (OID: 0.4.0.127.0.7.1.1.4.1.2) - ecdsa-plain-SHA256: (OID: 0.4.0.127.0.7.1.1.4.1.3) - ecdsa-plain-SHA384: (OID: 0.4.0.127.0.7.1.1.4.1.4) - ecdsa-plain-SHA512: (OID: 0.4.0.127.0.7.1.1.4.1.5) - ecdsa-plain-RIPEMD160: (OID: 0.4.0.127.0.7.1.1.4.1.6)

4.6 Unit LDS_I – EF CardAccess

This unit contains all mandatory tests regarding the presence and coding of PACE-related SecurityInfo structures in EF CardAccess.
If the EF CardAccess contains multiple instances of the same element type (PACEInfo, PACEDomainParameterInfo), the corresponding test cases have to be performed for each element. A test case is only rated as a PASS if all passes of a test case are performed without any failure, so that all test runs for one test case lead to a single result.

4.6.1 Test case LDS_I_01

Purpose	Test the ASN.1 encoding of the SecurityInfos
Version	2.0
References	[R7] TR-SAC §3.1
Profile	PACE
Preconditions	1. EF CardAccess MUST have been read from the eMRTD
Test scenario	1. Check the EF CardAccess file data 2. Check the SecurityInfos.
Expected results	1. The data content of the EF CardAccess MUST be encoded according to the SecurityInfos syntax definition 2. - At least one PACEInfo using a standardized domain parameter MUST be present. - SecurityInfos may contain additional entries indicating support for other protocols. The inspection system may discard any unknown entry. - The data structure PACEDomainParameterInfo is REQUIRED if the eMRTD chip provides proprietary domain parameters for PACE, and MUST be omitted otherwise

4.6.2 Test case LDS_I_02

Purpose	Test the ASN.1 encoding of the PACEInfo
Version	2.0
References	[R7] TR-SAC §3.1
Profile	PACE
Preconditions	1. EF CardAccess MUST have been read from the eMRTD 2. The Card Access content is parsed and this test is repeated for each PACEInfo element containing an OID as defined in the SAC specification [R7].
Test scenario	1. The PACEInfo element must follow the ASN.1 syntax definition in the SAC [R7] specification 2. The algorithm identifier MUST be one of the following: – id-PACE-DH-GM-3DES-CBC-CBC (OID : 0.4.0.127.0.7.2.2.4.1.1) – id-PACE-DH-GM-AES-CBC-CMAC-128 (OID : 0.4.0.127.0.7.2.2.4.1.2) – id-PACE-DH-GM-AES-CBC-CMAC-192 (OID : 0.4.0.127.0.7.2.2.4.1.3) – id-PACE-DH-GM-AES-CBC-CMAC-256 (OID : 0.4.0.127.0.7.2.2.4.1.4) – id-PACE-ECDH-GM-3DES-CBC-CBC (OID : 0.4.0.127.0.7.2.2.4.2.1) – id-PACE-ECDH-GM-AES-CBC-CMAC-128 (OID : 0.4.0.127.0.7.2.2.4.2.2) – id-PACE-ECDH-GM-AES-CBC-CMAC-192 (OID : 0.4.0.127.0.7.2.2.4.2.3) – id-PACE-ECDH-GM-AES-CBC-CMAC-256 (OID : 0.4.0.127.0.7.2.2.4.2.4) – id-PACE-DH-IM-3DES-CBC-CBC (OID : 0.4.0.127.0.7.2.2.4.3.1) – id-PACE-DH-IM-AES-CBC-CMAC-128 (OID : 0.4.0.127.0.7.2.2.4.3.2) – id-PACE-DH-IM-AES-CBC-CMAC-192 (OID : 0.4.0.127.0.7.2.2.4.3.3) – id-PACE-DH-IM-AES-CBC-CMAC-256 (OID : 0.4.0.127.0.7.2.2.4.3.4) – id-PACE-ECDH-IM-3DES-CBC-CBC (OID : 0.4.0.127.0.7.2.2.4.4.1) – id-PACE-ECDH-IM-AES-CBC-CMAC-128 (OID : 0.4.0.127.0.7.2.2.4.4.2) – id-PACE-ECDH-IM-AES-CBC-CMAC-192 (OID : 0.4.0.127.0.7.2.2.4.4.3) – id-PACE-ECDH-IM-AES-CBC-CMAC-256 (OID : 0.4.0.127.0.7.2.2.4.4.4) 3. The version MUST be encoded as INTEGER and MUST be 2 4. Check the ParameterId matches the domain parameters: The ParameterId if present MUST be one of the following (see table 6 of [R7] – ‘00’ if 1024-bit MODP Group with 160-bit Prime Order Subgroup is used.

	– ‘01’ if 2048-bit MODP Group with 224-bit Prime Order Subgroup is used. – ‘02’ if 2048-bit MODP Group with 256-bit Prime Order Subgroup is used. – ‘08’ if NIST P-192 is used – ‘09’ if BrainpoolP192r1 is used – ‘10’ if NIST P-224 is used – ‘11’ if BrainpoolP224r1 is used – ‘12’ if NIST P-256 is used – ‘13’ if BrainpoolP256r1 is used – ‘14’ if BrainpoolP320r1 is used – ‘15’ if NIST P-384 is used – ‘16’ if BrainpoolP384r1 is used – ‘17’ if BrainpoolP512r1 is used – ‘18’ if NIST P-521 is used – Above ‘32’ (included) for proprietary domain parameters
Expected results	1. true 2. true 3. true 4. true

4.6.3 Test case LDS_I_03

Purpose	Test the ASN.1 encoding of the PACEDomainParameterInfo
Version	2.0
References	[R7] TR-SAC §3.1
Profile	PACE
Preconditions	1. EF CardAccess MUST have been read from the eMRTD 2. The Card Access content is parsed and this test is repeated for each proprietary PACEDomainParameterInfo element. The test is skipped if no PACEDomainParameterInfo object is found.
Test scenario	1. The PACEDomainParameterInfo element must follow the ASN.1 syntax definition in the SAC [R7] specification 2. The protocol identifier MUST be one of the following: – id-PACE-DH-GM (OID : 0.4.0.127.0.7.2.2.4.1) – id-PACE-ECDH-GM (OID : 0.4.0.127.0.7.2.2.4.2) – id-PACE-DH-IM (OID : 0.4.0.127.0.7.2.2.4.3) – id-PACE-ECDH-IM (OID : 0.4.0.127.0.7.2.2.4.4) 3. The algorithm identifier MUST match to the key agreement protocol and be one of the following: – dhpublicnumber (OID: 1.2.840.10046.2.1) – ecPublicKey (OID: 1.2.840.10045.2.1) 4. The parameters MUST follow PKCS #3 (DH) or KAEG specification (ECDH). For DH verify that – $0 < g < p$, that is both should be positive and g should be less than p. – If private value length l is present, verify that $l > 0$ and $2^{l-1} < p$. In case of ECDH verify that – prime $p > 2$ – curve parameter $0 \leq a < p$ – curve parameter $0 \leq b < p$ – $4a^3 + 27b^2 \neq 0$ – base point G is on the curve, with both coordinates in range 0 … p – 1 – Cofactor $f > 0$ – order r of base point $r > 0$, $r \neq p$ – $r * f \leq 2p$ – the generator point is encoded in uncompressed format according to [R8], i.e ‘04||x||y’ 5. If a ParameterId is present in the PACEDomainParameterInfo element, there MUST be at least one PACEInfo element with this ParameterId. 6. If a ParameterId is present in the PACEDomainParameterInfo element, it must be larger than 31.
Expected	1. true

results	2. true 3. true 4. true 5. true 6. true

4.7 Unit LDS_J – Data Group 15

This unit contains all mandatory tests regarding the coding of data group 15. DG15 contains the public key information required for the Active Authentication mechanism.

4.7.1 Test Case LDS_J_01

Purpose	This test checks the template tag that the encoded EF.DG15 element starts with.
Version	2.0
References	[R1] 9303-3 vol 2 – section IV - 8
Profile	AA
Preconditions	1. EF.DG15 has been retrieved from the eMRTD
Test scenario	1. Check the very first byte of the EF.DG15 element
Expected results	1. First byte MUST be '6F'

4.7.2 Test Case LDS_J_02

Purpose	This test checks the encoding of EF.DG15 element length.
Version	2.0
References	[R1] 9303-3 vol 2 – section IV - 8
Profile	AA
Preconditions	1. EF.DG15 has been retrieved from the eMRTD
Test scenario	1. Analyze the encoding of the bytes that follow the template tag 2. Verify the length of the EF.DG15 object
Expected results	1. The bytes that follow the template tag MUST contain a valid length encoding (According to ASN.1 encoding rules). 2. The encoded length MUST match the size of the given EF.DG15 object.

4.7.3 Test Case LDS_J_03

Purpose	This test checks the DER-TLV encoding of the "Subject Public Key Info" present in EF.DG15.
Version	2.0
References	[R1] 9303-3 vol 2 – section IV – 8 RFC-3280
Profile	AA
Preconditions	1. EF.DG15 has been retrieved from the eMRTD 2. EF.DG15 contains a SubjectPublicKeyInfo value under the '6F' tag
Test scenario	1. Search for the AA Public Key Info (Tag '30') inside EF.DG15. 2. Check the DER-TLV encoding of the AA Public Key Info 3. Check the value of the encoded AA Public Key Info
Expected results	1. Tag '30' MUST be present. 2. The AA Public Key Info MUST be DER-encoded. 3. The AA Public Key Info MUST follow the encoding of the Subject Public Key Info specified in RFC-3280.

4.7.4 Test Case LDS_J_04

Purpose	This test checks that the algorithm indicated for the Public Key in EF.DG15 is one of the algorithms specified in [R1].
Version	2.0
References	[R1] 9303-3 vol 2 – section IV – 8 RFC-3279
Profile	AA
Preconditions	1. EF.DG15 has been retrieved from the eMRTD 2. EF.DG15 contains a SubjectPublicKeyInfo value under the '6F' tag 3. The SubjectPublicKeyInfo holds a sequence TLV
Test scenario	1. Search for the Algorithm Identifier (Tag '30') inside the AA Public Key Info. 2. Check the DER-TLV encoding of the Algorithm Identifier 3. Check the value of the Algorithm Identifier 4. Check the value of the algorithm indicated in the Algorithm Identifier
Expected results	1. Tag '30' MUST be present and MUST occur only once. 2. The Algorithm Identifier MUST be DER-encoded. 3. The Algorithm Identifier MUST follow the ASN.1 encoding specified in RFC-3280. 4. The Public Key Algorithm indicated in the Algorithm Identifier MUST be one of the algorithms indicated in RFC-3279 (i.e. the OID of the algorithm MUST be rsaEncryption or id-ecPublicKey).

4.7.5 Test Case LDS_J _05

Purpose	This test checks the encoding of the Subject Public Key in the AA Public Key Info in EF.DG15.
Version	2.0
References	[R1] 9303-3 vol 2 – section IV – 8 RFC-3280 RFC-3279
Profile	AA
Preconditions	1. EF.DG15 has been retrieved from the eMRTD 2. EF.DG15 contains a SubjectPublicKeyInfo value under the '6F' tag 3. The SubjectPublicKeyInfo holds a sequence TLV
Test scenario	1. Search for the Subject Public Key (Tag '03') inside the AA Public Key Info. 2. Check the DER-TLV encoding of the Subject Public Key 3. Check that the data bits from the bit string code a valid Public Key for the algorithm indicated in the Subject Public Key Info data element. 4. Checks that the length of the encoded Public Key meets the minimum size recommendations.
Expected results	1. Tag '03' MUST be present and MUST occur only once. 2. The Subject Public Key MUST be encoded as a bit-string. 3. The data bits from the bit string MUST code a valid Public Key for the algorithm indicated in the Subject Public Key Info data element. 4. An RSA Public Key MUST have a length of at least 1024 bits, an EC Public Key MUST have a length of at least 160 bits.